Microcomputer Control System for Programmable Logic Array Burn-in

by

Mousa Isa Gaddourah

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGIEERING

October, 1980

Order Number 1355746

Microcomputer control system for programmable logic array burn-in

Gaddourah, Mousa Isa, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1980

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MICROCOMPUTER CONTROL SYSTEM FOR PROGRAMMABLE LOGIC ARRAY BURN-IN

BY

MOUSA ISA GADDOURAH

THESIS

PRESENTED TO THE FACULTY OF THE COLLEGE OF GRADUATE STUDIES

UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

OCTOBER 1980

UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

This thesis, written by

MOUSA ISA GADDOURAH

under the direction of his Thesis Committee, and approved by all its members, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Dr. Reda S. Al-Thiga
Dean, College of Graduate Studies
Date: 14/11/4

Dr. Samir J. Al-Bader
Department Chairman

THESIS COMMITTEE

Dr. Thaddeus J. Kobylarz
Chairman

Dr. Reda S. Al-Thiga
Member

Dr. Ahmed S. Farag
Member


بسم الله الرحمن الرحيم

This thesis is dedicated to my family

ACKNOWLEDGEMENTS

Acknowledgement is due to the University of Petroleum and Minerals for support of this research.

I wish to express my appreciation to Professor T.J. Kobylarz who served as my major advisor. I also wish to thank the other members of my Thesis Committee, Dr. Al-Thiga and Dr. Farag. Finally, I would like to thank the Electrical Engineering Laboratory Technicians, especially Mr. Frazi, Mr. Cecil and Mr. Bahgat for their enormous help.

Thanks are also due to Mr. Jawad A. Qureshi and Mr. Khalid Butt for doing an excellent job in typing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

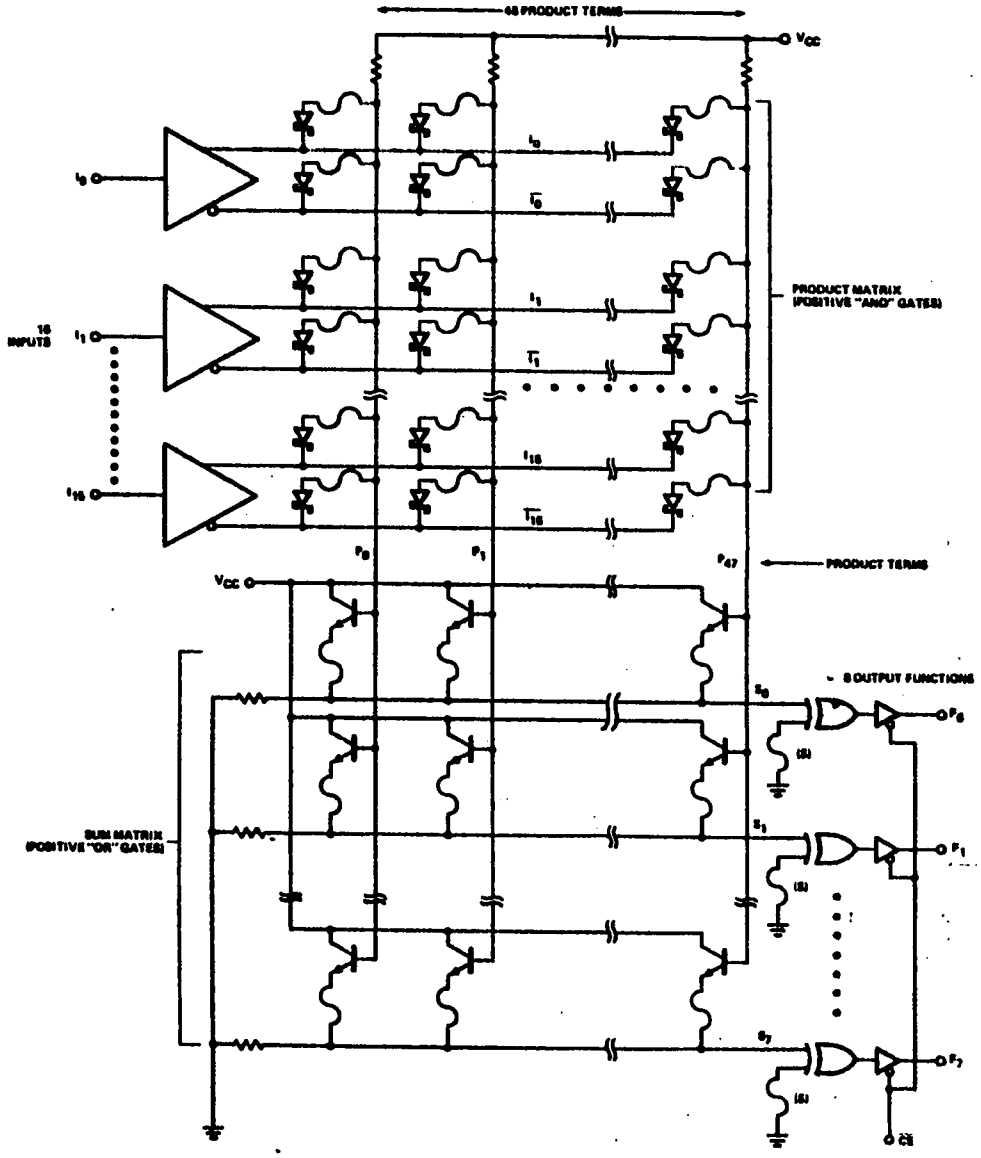
The programmable logic array (PLA) is a large scale integration (LSI) device. It can be considered as a two-level AND-OR logic circuit. The programming of this device is done in three major steps. These steps are: 1. Output burn-in for an inverted output, 2. Product burn-in to select the state of the inputs of the AND gates chosen, 3. Sum burn-in to select the OR gates to be left connected to the used AND gates. A prototype microcomputer control system to program the PLA was developed. The system is principally software dependent and uses hexadecimal notation as input data. The system is also made flexible for future modifications.

I. INTRODUCTION

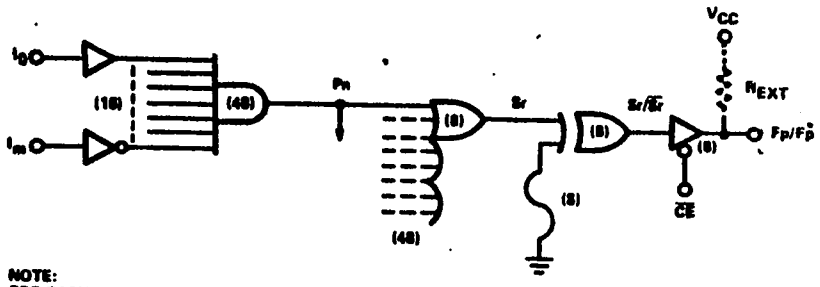
1.1 DEFINITION

Large Scale Integration (LSI) circuits have greatly reduced the cost of digital systems because of the large quantity of elements. One very useful LSI circuit (chip) is the Programmable Logic Array (PLA). The PLA is basically a two level logic system. Typically the first level consists of 48 AND gates, while the second level consists of 8 OR gates |1|. It usually has 16 inputs and 8 outputs. In the first level, each AND gate can be connected to any combination of the 16 inputs, including their complements. This level generates a possible number of product terms (P-terms) of a switching function, as the AND gates. In the second level, any combination of the outputs of the AND gates can be connected to the OR gates to generate the sum terms. An output function can be inverted if desired, by means of an exclusive OR gate. Figure 1.1 shows the PLA diagram and its simplified Typical logic path |1|. Figure 1.2 shows the pin configuration of the PLA used in this work (Signetics 82S100) |1|. Table I lists two other PLAs currently available from manufacturers.

The programming of the PLA is done by burning undesired fuses interconnecting the elements within the chip. A major constraint in the PLA is the number of AND gates. Hence, the programmer, often needs to reduce the number of AND gates, which are implied by initial function specifications, by applying suitable logic analysis techniques |2|.



a. PLA Block Diagram.



NOTE:
FOR EACH OF THE 8 OUTPUTS, EITHER THE FUNCTION F_p (ACTIVE HIGH) OR $\overline{F_p}$ (ACTIVE LOW) IS AVAILABLE, BUT NOT BOTH. THE REQUIRED FUNCTION POLARITY IS USER PROGRAMMABLE VIA FUSE (S).

$P_n = I_0 \overline{I_1}\,\overline{I_2}\, I_3 \ldots\ldots \overline{I_m}$
$S_r = P_0 + P_1 + P_2 + \ldots\ldots P_n$
$\overline{S_r} = \overline{P_0} \bullet \overline{P_1} \bullet \overline{P_2} \bullet \ldots\ldots \overline{P_n}$
$F_p = (\overline{CE}) + (S_r) = (\overline{CE}) + (P_0 + P_1 + P_2 + \ldots\ldots P_n)$ ● S = SHORT
$\overset{\circ}{F_p} = (\overline{CE}) + (\overline{S_r}) = (\overline{CE}) + (\overline{P_0} \bullet \overline{P_1} \bullet \overline{P_2} \bullet \ldots\ldots \overline{P_n})$ ● S = OPEN

b. PLA Typical Logic Path.

Figure 1.1 PLA Block Diagram and Logic Path.

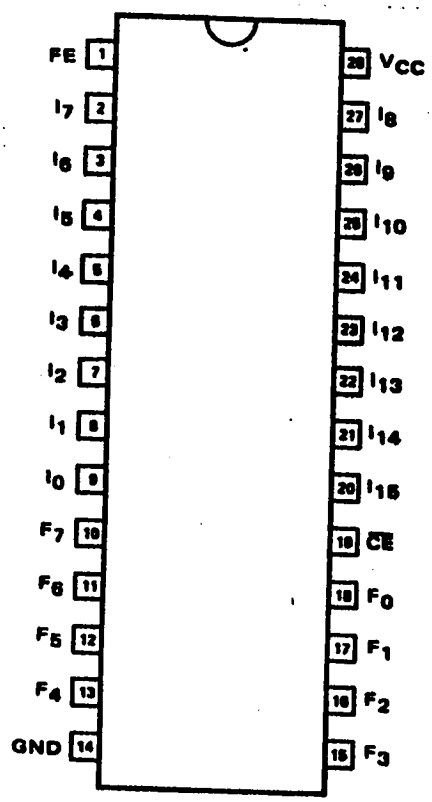


Figure 1.2. PLA Pin Configuration .

TABLE I. Other PLAs Currently Available From Manufacturers.

PLA Designation	Manufacturer	No. of inputs	No. of outputs
MI5200	MMI	14	8
DM7575	National semiconductor	14	8

The PLA can replace a large number of Small Scale Integration (SSI) circuits in a logic design. By doing so it affords a great economical advantage. Wiring and drilling holes in printed circuit (PC) boards and soldering of integrated circuits is especially arduous and time consuming.

To illustrate the usefullness of the PLA, consider an example of a two-level logical circuit which decodes 4 bits into hexadecimal characters in Arabic and English. The circuits require 5 inputs and 7 outputs. A total number of 31 AND gates at the first level are needed. If PLA's are used to realize this circuit, one PLA is needed. For (SSI) 8 IC's are needed for the first level, each has an AND gate of 5 inputs. Including OR gates and inverters, more than 50 chips are needed. Hence, a considerable reduction in cost is achieved. Furthermore, the complexity of the connections is averted and the PC board area is reduced.

1.2 ADVANTAGES OF PLA OVER ROM

A "read only memory" (ROM) can also be considered as a programmable logic array. However, the ROM has certain disadvantages. Table II compares a 16 input PLA with the largest ROM which could be found |1,3|. In a ROM, all internal words are reached by a fixed decoder internal to the device. The size of this decoder, as well as the storage matrix, doubles for each additional address input. Hence, a 65, 536 x 8 bits ROM would be needed to provide equivalent function of a 16 input, 8 output PLA |4|. By

TABLE II. PLA Vs ROM Data .

Parameter	Typical PLA	Largest ROM	Units
No. of inputs	16	10	
No. of outputs	8	8	
No. of AND gates	48	1024	
Typical access time	50	47	ns
Typical power dissipation	550	700	mw
Price (1979)	31	20	$

means of simple calculations, using Table II, 64 (= 2^6) ROM's will be required for 16 input functions. The total price of these ROMs will be about $ 1280 and the power required is 44.8 watts. A glance at these figures indicates that ROM's are not an alternative to PLA's. Another advantage of PLA's over ROM's is that the PLA stores only the states required to generate the output functions in the form of logic equations |5|, and in particular, ignores "don't-care" states |6|. By comparison, all addresses must be stored for the ROM, whether needed to generate the output functions or not. Hence, the rest of the ROM, if only few bits are needed, is unused and would be programmed with zero's (or ones). While in the PLA, the remainder would be available for other logic. In common with the ROM type of designs, PLA based products have much greater design flexibility than random logic systems. Instead of being restricted by available SSI and MSI functions, integrated circuit (IC) counts, power and fan-out restrictions, and the rigidness of PC board layouts, the PLA designer can modify his system by changing the PLA program |7|.

II. LITERATURE REVIEW

2.1 APPLICATIONS

The PLA circuit configuration is in the form of two-rail AND-OR logic. Hence, the PLA can be used for all functions definable by a truth table. This feature provides a wide range of applications for the PLA. Figure 2.1 illustrates some of the applications |1|.

In address mapping applications, the PLA can be looked at as a large read only memory (ROM). For many situations it provides the capability to scan an address field 65,536 words deep |1|.

The PLA is a viable alternative for improved character generation. It provides the required serial bit rate and the characters are not stored as fixed sized dot pattern. They are composed of substructures that can be described by terms in logic equations. This will not increase PLA storage requirements as much as it would with ROM storage |7|.

The combinational logic capabilities of a PLA can be used in sequential logic systems. In this case, some of the outputs of the PLA are fedback through storage flip-flops (FF) to the input as shown in Fig. 2.2. A system clock sequences the system, and each successive output is dependent on the input and the previous state |7|.

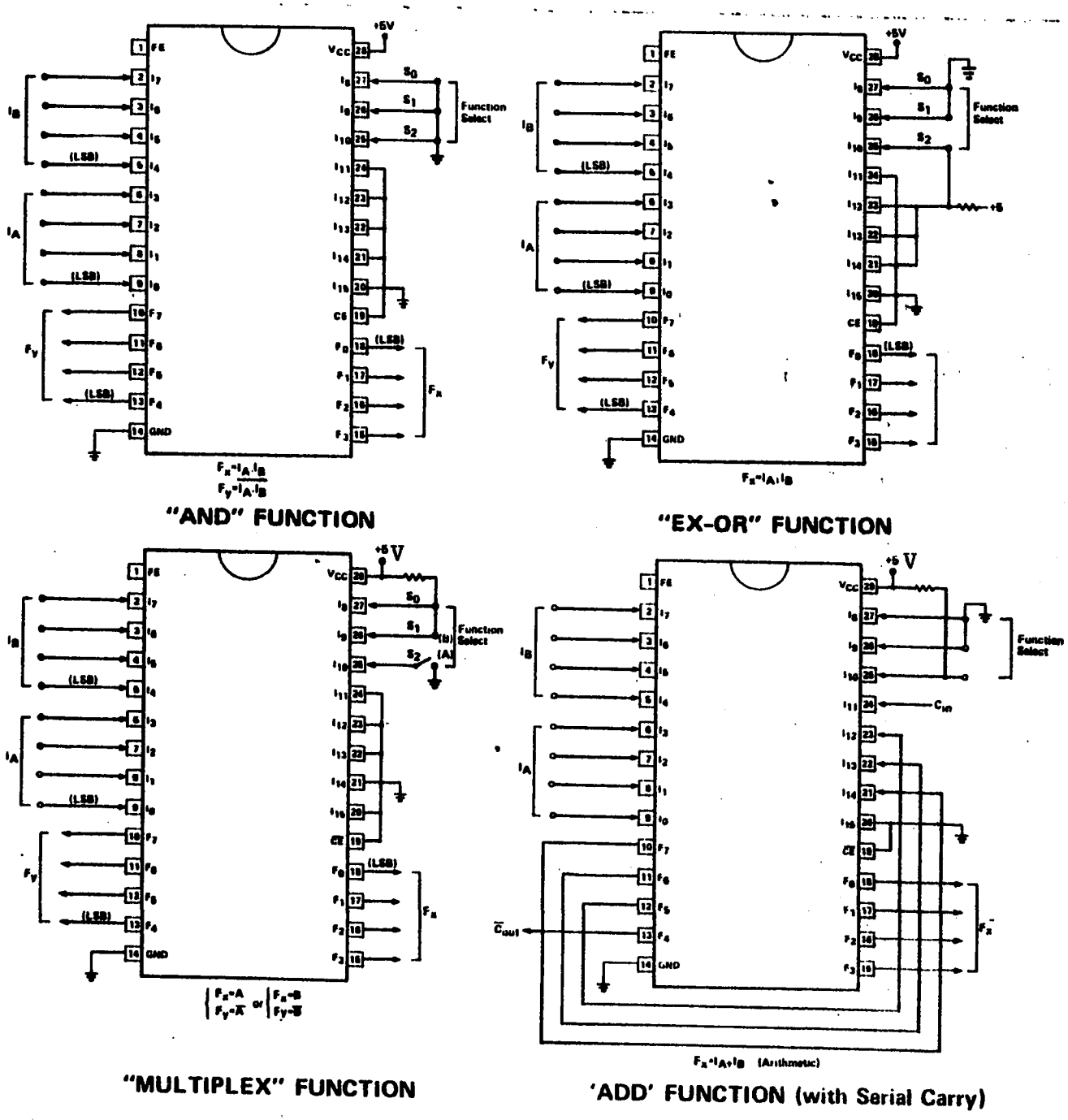


Figure 2.1 Sample PLA Applications .

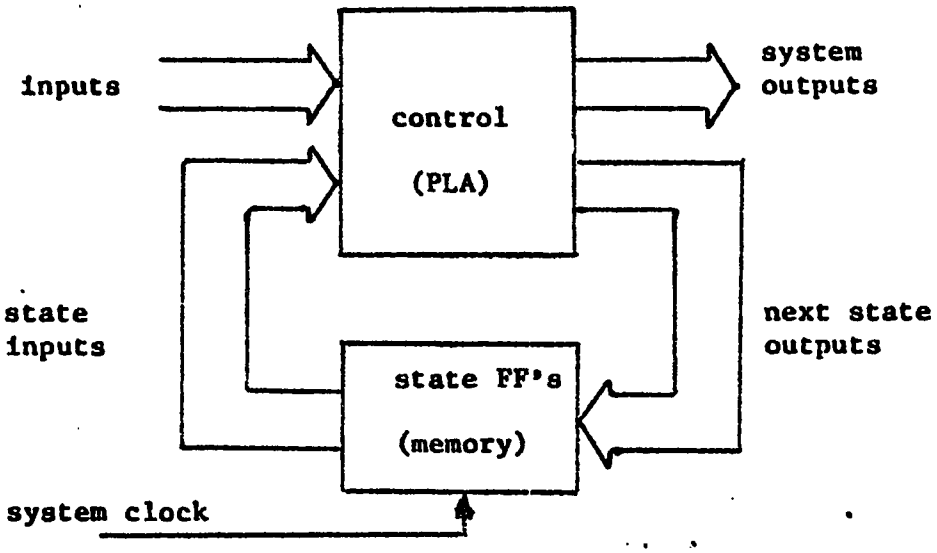


Figure 2.2. PLA Usage for Sequential Logic.

The control of a CO_2 laser system is a special application of the PLA when used in sequential logic systems. Figure 2.3 illustrates PLA controller capability when controlling a section of a CO_2 laser system used to scribe sapphire wafers |7|. This system controls an X-Y stage for positioning wafers and controls the firing of the laser during scribing operation |7|.

Still another application of the PLA is to implement a decision table. The structure of the PLA and the decision table are exactly analogous - the condition entries drive the address inputs, and the data-output bits represent the desired output actions. The PLA is connected as a peripheral device to the system's data bus as shown in Fig. 2.4. When the processor reads the PLA, it executes the tasks specified by true bits |8|. This application can be extended to using a PLA as the microprogram storage within a control unit of a computer.

One of N(1/N) detectors, as a special case of m/N decoder networks, is useful in a variety of applications in computers, data communications, and fault monitor systems. Since each PLA can examine 16 terminals, five are sufficient to service 80 status monitor lines. Each PLA utilizes 17 P-terms to detect the presence of zero, 1/16, or more connections via outputs (X) and (Y), as shown in Fig. 2.5. An additional PLA is necessary to examine a total of 10 partial X and Y results from the first level devices, and to give

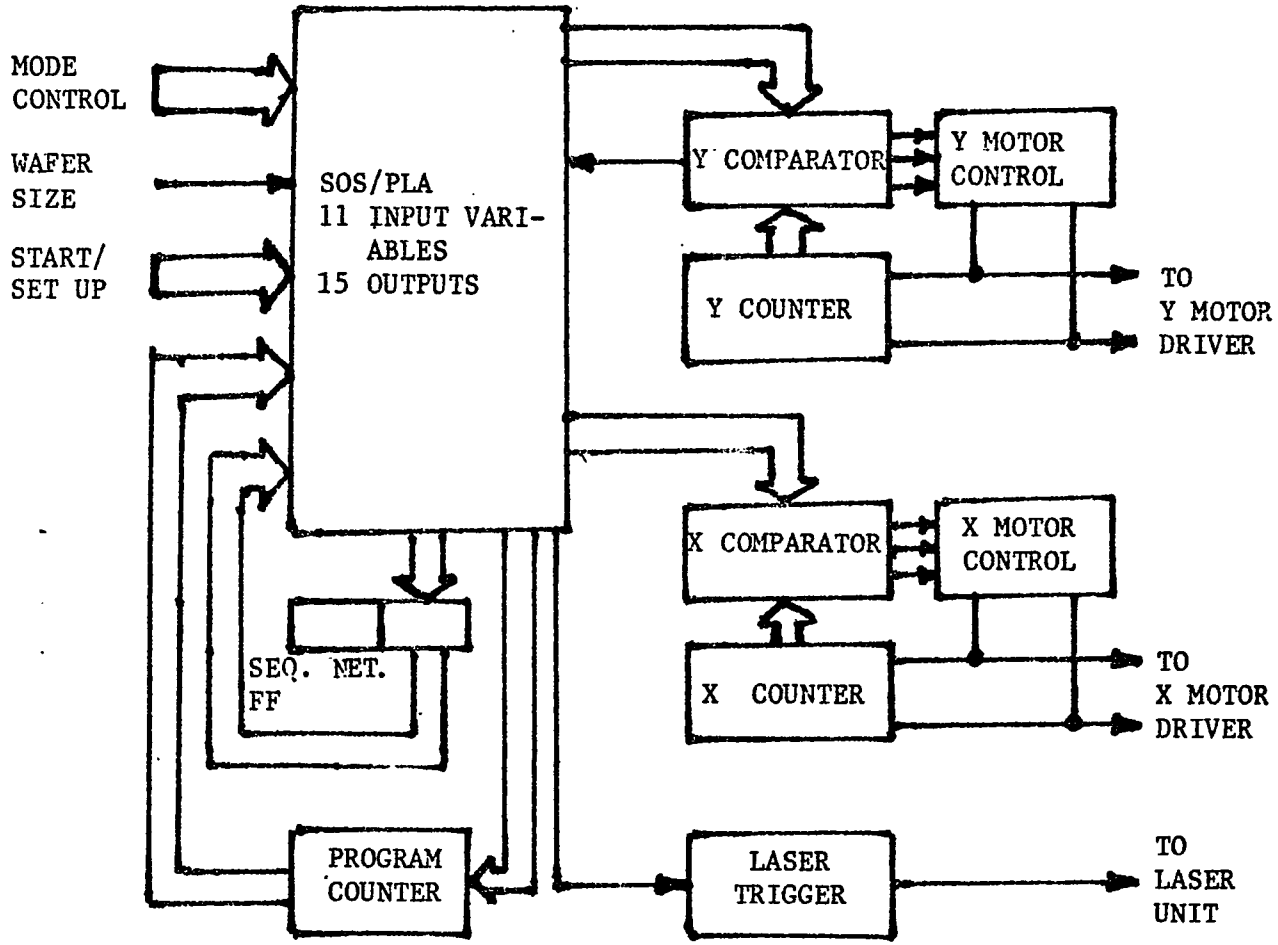


Figure 2.3 A Laser Scribe System Using the PLA as a Controller.

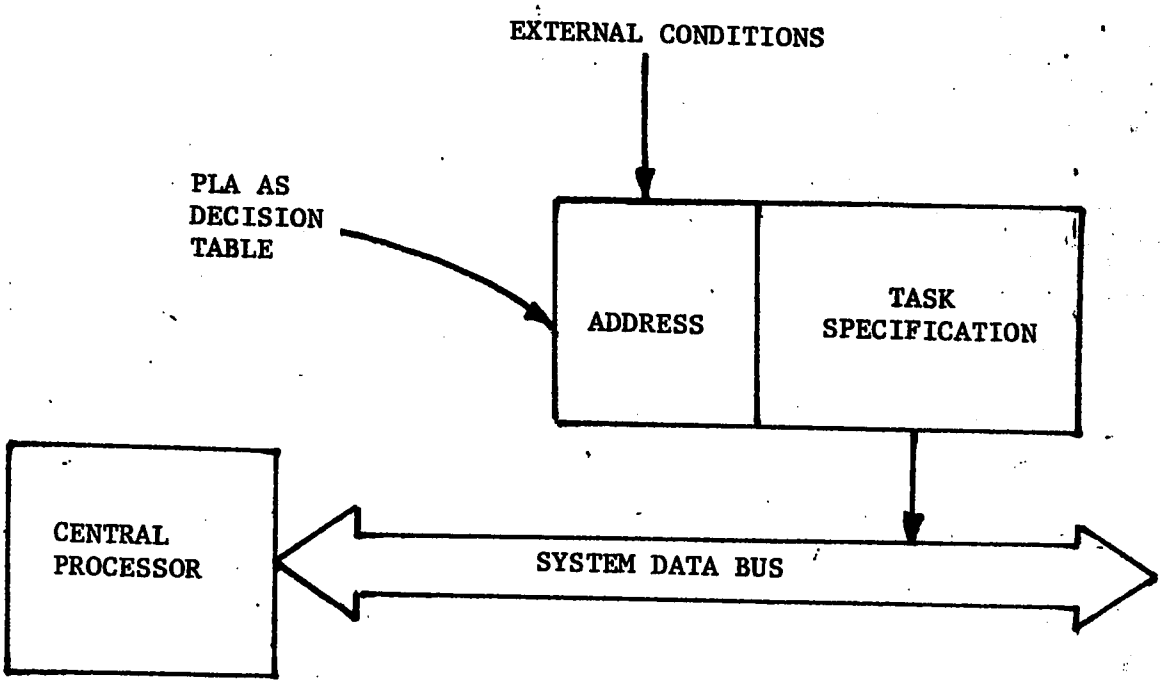
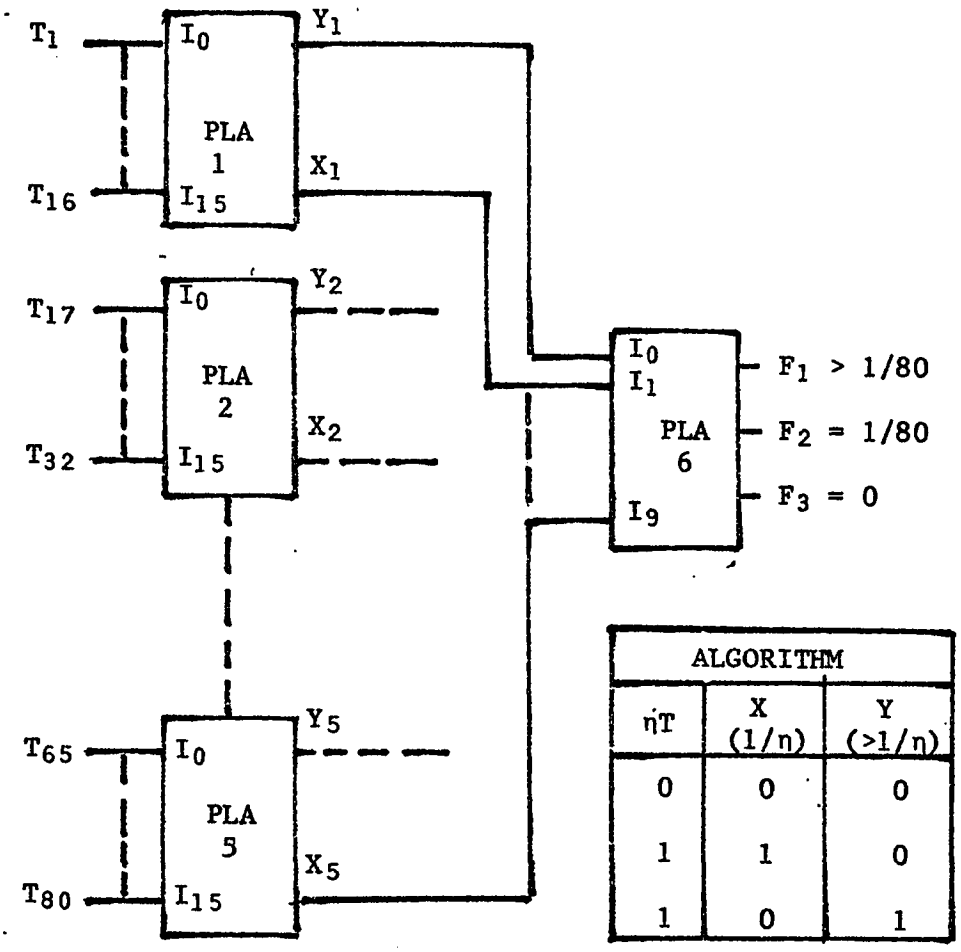


Figure 2.4. PLA As a Decision Table .



ALGORITHM		
ηT	X (1/η)	Y (>1/η)
0	0	0
1	1	0
1	0	1

Figure 2.5. 1/80 Detector with PLA's .

final indication of the number of selected channels [9].

In all these applications, the PLA will lower system size and improve performance. Above all, a considerable decrease in cost is achieved by the PLA.

2.2 BURN-IN OF THE PLA

As mentioned before, the programming of the PLA is a "burn-in" process. The programmer will burn unwanted fuses of undesired elements. To carry out this process, the manufacturer has suggested two alternatives [1]:

1. The manufacturer will perform burn-in provided that the customer sends function information in a certain format.

2. The customer will perform burn-in according to instructions provided by the manufacturer. Tabulated below is a sample of a portion of instructions provided by Signetics Inc. [1]. Details related to these instructions is described later.

Output Polarity:

1. Set FE (pin 1) to V_{FEL},

2. Set V_{CC} (pin 28) to V_{CCL},

3. Set $\overline{CE}$ (pin 19) and I_0 through I_{15} to V_{IH},

4. Apply V_{OPH} to the appropriate output and remove

after a period t_p,

5. Repeat step 4 to program other outputs.

Figure 2.6 shows a circuit for a manual fuser (burn-in) circuit |1|.

When examining the first alternative manual burn-in, the reader will find that they are time and money consuming. In the latter, the human factor will affect the results. An extensive training program is necessary in order to have adequate results. Furthermore, extensive labor will preclude the PLA advantages because of long and costly programming procedure. Unfortunately, commercial equipment for automatic burn-in is unavailable. It is therefore necessary to design and fabricate such a system |10|.

In order to easily program a PLA, two facilities should be available |2|:

1. A minimization program which will reduce the number of AND gates for given functions. This is needed in order to realize a specific design with the smallest quantity of PLA elements.

2. A microcomputer system to perform the burn-in. A microcomputer system is essential because of the virtual impossibility for manually controlled burn-in.

The minimization program will be discussed in section 2.4. The microcomputer system design is the subject of this research.

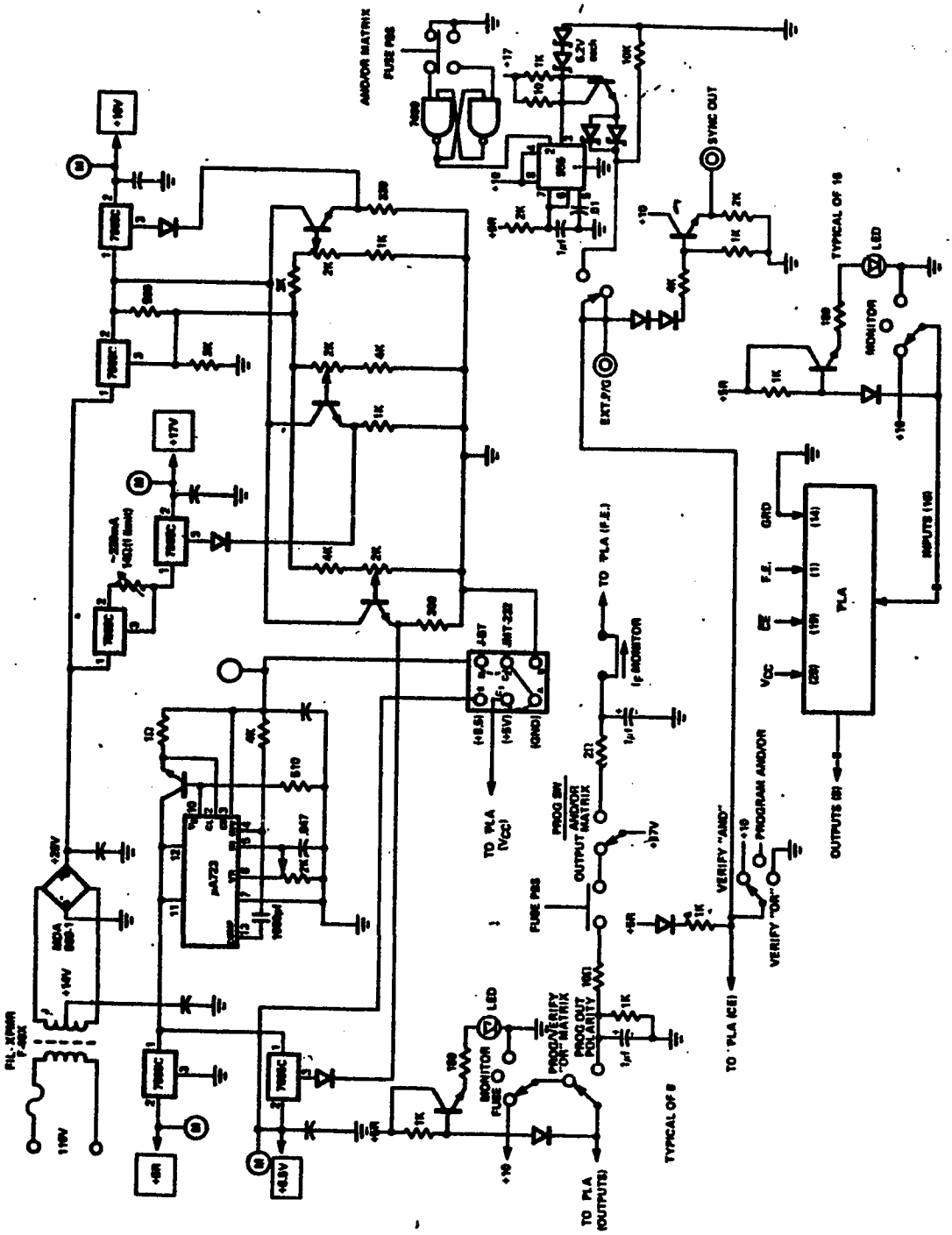


Figure 2.6 PLA Manual Fuser Circuit.

2.3 ROM PROGRAMMERS

The ROM programmers determined available in the market can not be used to program the PLA. The reason is that PLA and ROMs differ in many aspects. They differ in number of inputs and outputs, burning voltage and current specification, and programming sequence. A paper was written |11| indicating that some initial work has been done to produce a PLA programmer, but the author did not specify which PLA is being considered.

2.4 MINIMIZATION PROGRAM

For economic use of the PLA, a minimization algorithm to reduce the number of AND gates has been developed. The developed algorithm uses a new idea of "partitioning", which exploits the property that a single output algorithm can perform multiple output minimization for a PLA. It also uses a new concept of an "exclusive" minterm of a function or product function to reduce the total number of prime implicants generated. A flow chart of the program is shown in Fig. 2.7. The program is written in Fortran IV, and uses decimal numbers for the representation of inputs and outputs. The program, as it exists now, can handle up to 14 inputs and 16 outputs if all the existing virtual memory on the IBM 370 is made available. In order to connect the minimization program

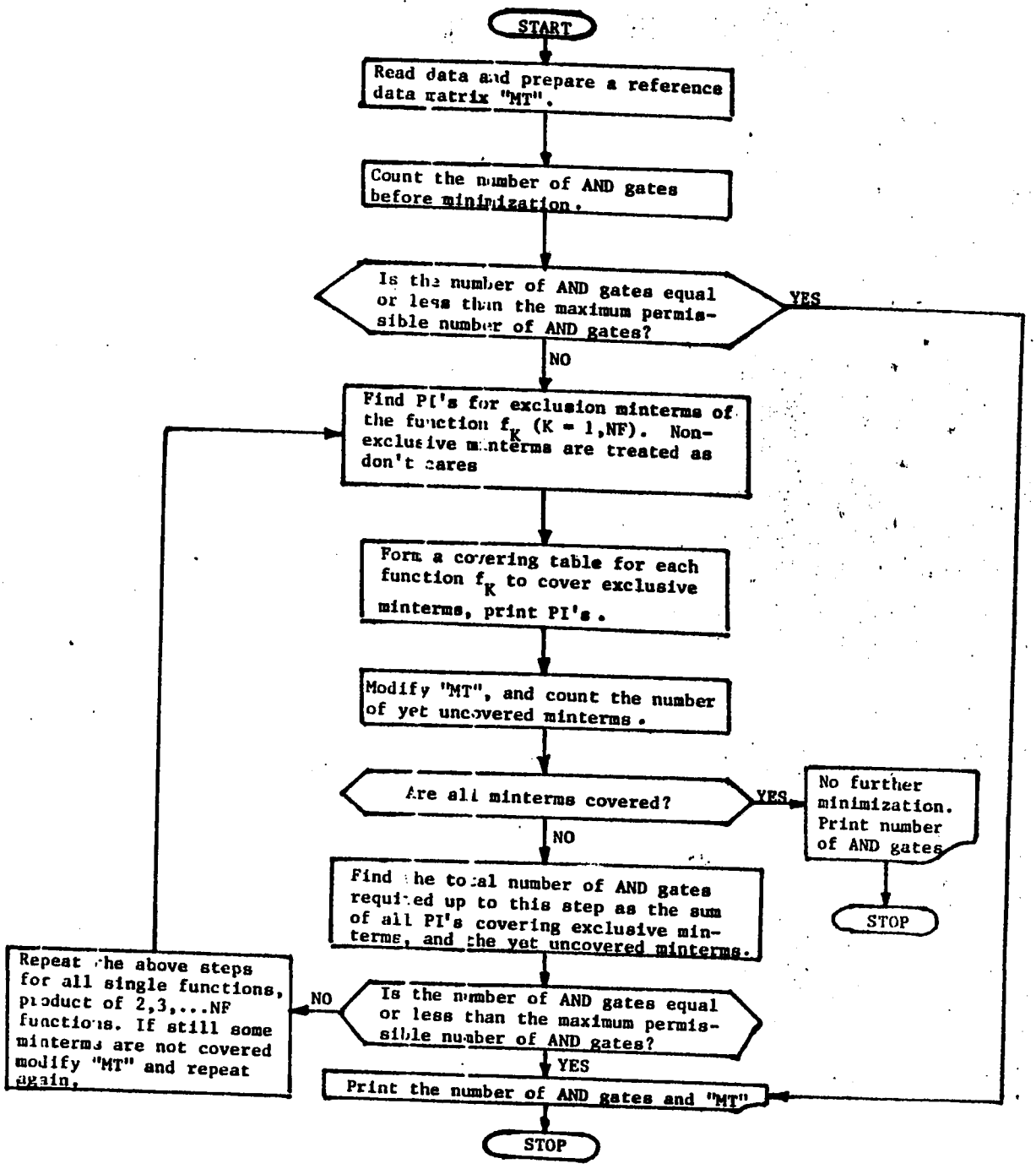


Figure 2.7. Flow Chart to Show Major Steps of the Algorithm.

to the microcomputer system, a subroutine should be added to the main program, to transfer the required information. This is an additional task pursued in the thesis work and a more detailed discussion will be given in section 4.2.3.

III. OVERVIEW AND MICROCOMPUTER DESCRIPTION

3.1 INTRODUCTION

Programming (burn-in) of the PLA is done by three major procedures:

1. Output,
2. Product,
3. Sum.

Fusing of Exclusive OR gates is performed during the Output procedure. Similarly, fusing of AND gates is performed during Product procedure, and finally fusing of OR gates during Sum procedure. Verification of each procedure is accomplished at the end of the specified procedure. The programming follows special instructions and accomodates the characteristics of the device as specified by the manufacturer. Appendix A describes specifications, instructions, and characteristics for programming the device |12|. A microcomputer controlled system is utilized to accomodate these requirements.

The major portions of the microcomputer system are as shown in Fig. 3.1 |2|. These are:

1. Microcomputer, which is used to control the hardware interface circuit after acquiring the data. A TTY is used to load data and initiate the programs.

2. Programming algorithm, which accepts data coded in a special format. The data can be from a minimization program |10|

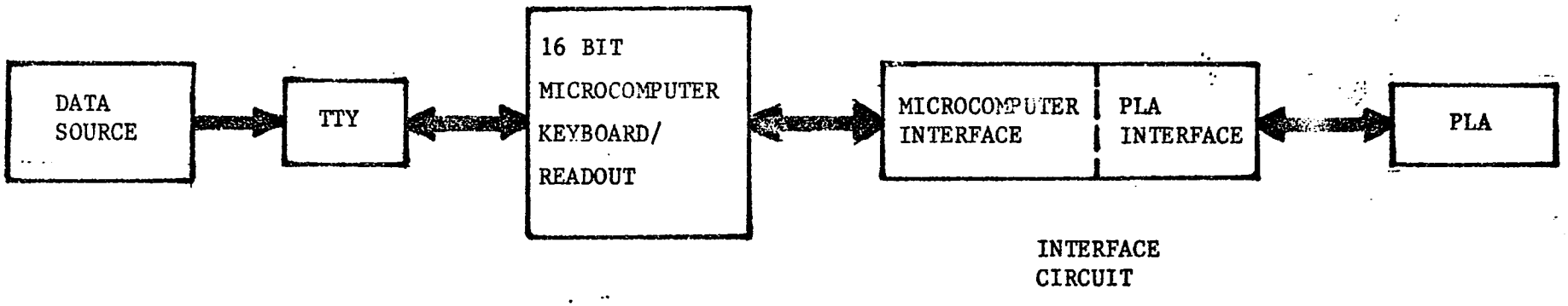


Figure 3.1. Microcomputer Control System Block Diagram.

or any other source. The algorithm is designed to meet the specified programming instructions. This aspect of the system is described in Chapter 4.

3. Bidirectional interface circuit, which is designed to accept control instructions from microcomputer and to meet PLA characteristics for programming. The bidirectional circuit is composed of the microcomputer interface and the PLA interface. These subsystems, are discussed in Chapters 5 and 6 respectively.

Considerations taken into account in the development of the system are:

1. The use of available hardware.
2. Minimization of hardware as much as possible.
3. Simplification and ease in debugging the system.
4. Development of fast and efficient software.
5. Minimization of data to be supplied to the system .
6. The use of easy and efficient format to code data from the minimization program or any other source.

3.2 MICROCOMPUTER DESCRIPTION

3.2.1 General Aspects

The microcomputer which was used is IPC-16A manufactured by National Semiconductor Corporation |13|. It is called a Pace Low Cost Development System (LCDS). The system provides capabilities for developing, testing, and debugging Pace hardware and software application design. The microcomputer operates on 8 or 16 bit

words and has 1K (1024), 16 bit words of RAM memory. Figure 3.2 shows a simplified block diagram of the LCDS |13|. A dual function 16 push-button keyboard and a 6 digit LED display facilitates direct control of the system. A variety of system monitor and control capabilities, as well as serial input/output subroutines, are permitted through a firmware program stored in the ROM's. The microcomputer can be used in conjuction with a TTY having either a 20 mA current loop or a RS-232 standard interface. Other electronic data terminals can be interfaced too. Three parallel 72-pin connection sockets are provided with the system. They are parallel input/output data terminals, with controlling strobes. Pertinent connection points are shown in Fig. 3.3. This port is used to interconnect the hardware section of the design.

3.2.2 Data Representation

Data is represented in "two's" complement integer notation |14|. Although it is a 16 bit system, it can be used as an 8 bit system with the highest order bit corresponding to the sign. The maximum range for a 16-bit number in this system is $7FFF_{16}$ ($+32767_{10}$) to 8000_{16} (-32768_{10}). Hexadecimal numbers are used to represent data from and to the user.

3.2.3 Registers

The system has 16, 16 bit registers. These are as follows:

1. Accumulators (AC0, AC1, AC2, AC3) are general-purpose registers used for performing arithmetic and logic operations, data transfers, skips, shifts, and rotates |13|. Normally, they are used as

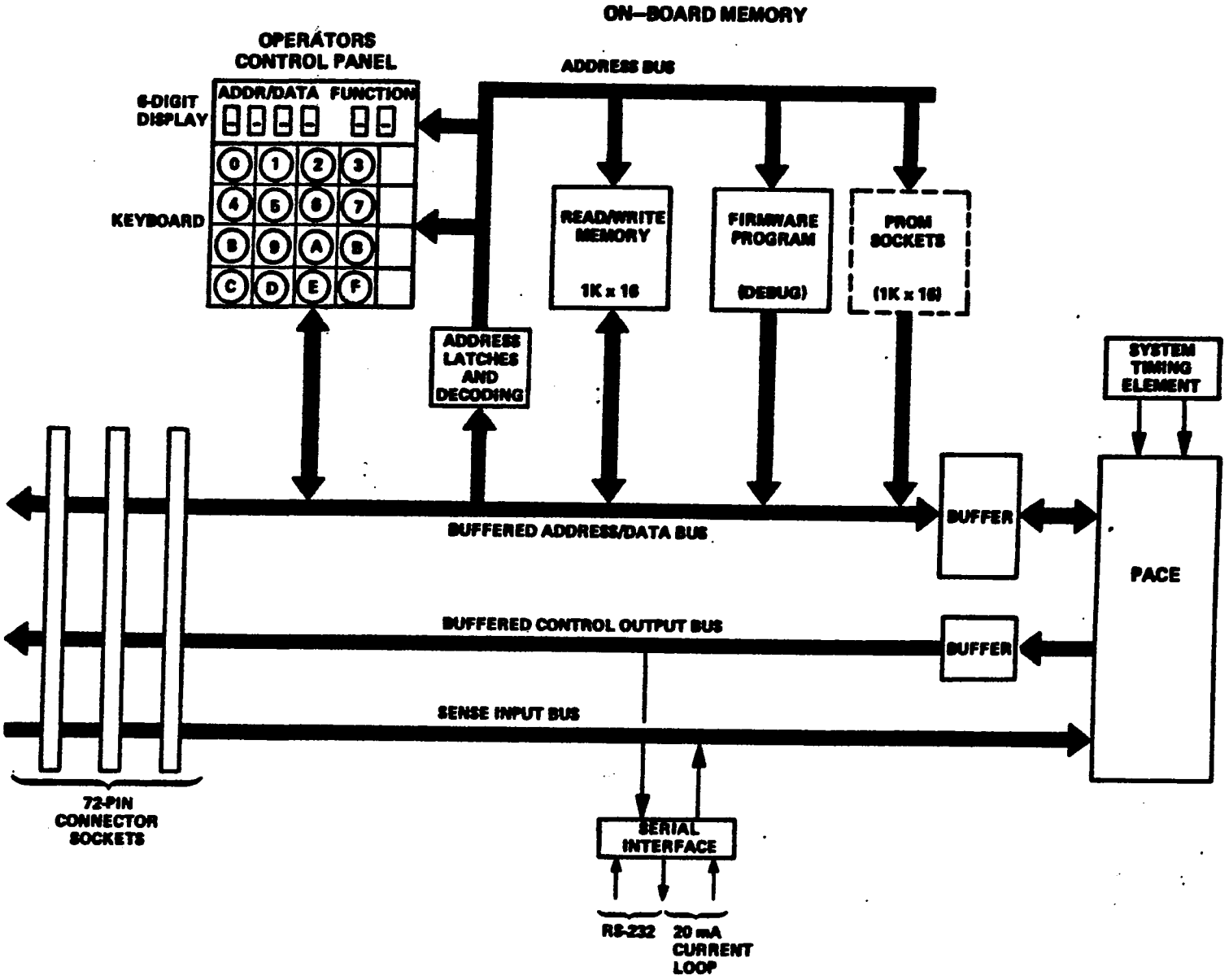


Figure 3.2. Simplified Block Diagram of the LCDS.

+5V	1	2	
	3	4	Ground (GND)
	5	6	
	7	8	
	9	10	
	11	12	
Clock (CLK)	13	14	Negative buffered address strobe (NBADS)
	15	16	
Buffered output data strobe (BODS)	17	18	Buffered input data strobe (BIDS)
	19	20	
	21	22	
	23	24	
	25	26	
	27	28	
	29	30	
	31	32	
	33	34	
	35	36	
Buffered data No.08(BD08)	37	38	BD00
BD09	39	40	BD01
BD10	41	42	BD02
BD11	43	44	BD03
	45	46	
BD12	47	48	BD04
BD13	49	50	BD05
BD14	51	52	BD06
BD15	53	54	BD07
	55	56	
	57	58	
	59	60	
	61	62	
	63	64	
	65	66	
	67	68	
	69	70	
	71	72	

(Note. Empty locations : No connection)

Figure 3.3. Pertinent Connection Points at the I/O Data Terminal.

follows:

AC0 - primary data-handling register

AC1 - secondary data-handling register

AC2 - indexed addressing of memory

AC3 - indexed addressing of memory and peripherals.

2. Status Flags Register (FR) provides storage for interrupt, arithmetic, control, and status flags |14|. The bit position of each flag is shown in Table III |14| along with their definitions.

3. Program Counter (PC) if used to store the address of the next instruction to be executed. The address of PC is always 4 greater than the address of the instruction being executed by 1.

4. Stack is used for data storage and status preservation, as well as for subroutines. It operates as first in/last out (FILO). Ten such registers exist.

3.2.4 Pace Instructions

The Pace system instruction set provides branch, ship, shift and rotate, memory data transfer, and other operations among the accumulators, memory and other registers |14|. The 46 instruction set is defined in Table IV with its hexadecimal codes |14|. The system accepts these instructions coded in machine language. The average time required to execute one instruction is about 12 μ sec. This fact is used later to achieve PLA programming specifications.

The sample program in Table V illustrates the use of some of the microcomputer instructions. The function of the program is

TABLE III . Description of Status and Control Flags .

Register Bit	Flag Name	Description	Flag code
0	'1'	Not used. Always high.	0000
1	IE1	Interrupt Enable level 1.	0001
2	IE2	Interrupt Enable level 2.	0010
3	IE3	Interrupt Enable level 3.	0011
4	IE4	Interrupt Enable level 4.	0100
5	IE5	Interrupt Enable level 5.	0101
6	OV	Overflows Flag is set to state of two's complement arithmetic overflow by arithmetic instructions.	0110
7	CY	Carry flag is set to state of binary or decimal carry output of adder by arithmetic instructions.	0111
8	LINK	LINK Flag is included in shift and rotate operations.	1000
9	IEN	Master Interrupt Enable Flag simultaneously inhibits all five of lowest priority interrupt levels.	1001
10	BYTE	Byte Flag selects 8-bit data length when high.	1010
11-14	F11-F14	General Purpose control flags.	1011 - 1110
15	'1'	Not functional. Always in logic 1 .	1111

TABLE IV . Microcomputer Instruction Set.

Description	Opcode Base	Source Statement	Instruction Format	Operation
Branch Instructions				
Branch On Condition*	4000	BOC cc,address	15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 1 0 0 cc | disp	If cc true, (PC) ← (PC) + disp, B
Jump	1800	JMP	0 0 0 1 1 0 | xr | disp	(PC) ← EA
Jump Indirect	9800	JMP @ address/disp(xr)	1 0 0 1 1 0 | xr | disp	(PC) ← (EA)
Jump to Subroutine	1400	JSR	0 0 0 1 0 1 | xr | disp	(STK) ← (PC), (PC) ← EA
Jump to Subroutine Indirect	9400	JSR @	1 0 0 1 0 1 | xr | disp	(STK) ← (PC), (PC) ← (EA)
Return from Subroutine	8000	RTS disp	1 0 0 0 0 0 0 0 0 0	(PC) ← (STK) + disp
Return from Interrupt	7C00	RTI	0 1 1 1 1 1 0 0 0 0	(PC) ← (STK) + disp, IEN = 1
Skip Instructions				
Skip if Not Equal	F000	SKNE r,	15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 1 1 1 1 r	If (ACr) ≠ (EA), (PC) ← (PC) + 1, B
Skip if Greater	9C00	SKG 0,	1 0 0 1 1 1	If (AC0) > (EA), (PC) ← (PC) + 1, B
Skip if AND is Zero	B800	SKAZ 0, address/disp(xr)	1 0 1 1 1 0 | xr | disp	If [(AC0) ∧ (EA)] = 0, (PC) ← (PC) + 1, B
Increment and Skip if Zero	8C00	ISZ	1 0 0 0 1 1	(EA) ← (EA) + 1; If (EA) = 0, (PC) ← (PC) + 1, B
Decrement and Skip if Zero	AC00	DSZ	1 0 1 0 1 1	(EA) ← (EA) − 1; If (EA) = 0, (PC) ← (PC) + 1, B
Add Immediate, Skip if Zero	7800	AISZ r,data	0 1 1 1 1 0 | r | data	(ACr) ← (ACr) + data; If (ACr) = 0, (PC) ← (PC) + 1
Memory Data-Transfer Instructions				
Load	C000	LD r,	15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 1 1 0 0 r	(ACr) ← (EA), EA = (ACxr) + disp
Load Indirect	A000	LD 0,@	1 0 1 0 0 0	(AC0) ← (EA), EA = ((ACxr) + disp)
Store	D000	ST r, address/disp(xr)	1 1 0 1 r | xr | disp	(EA) ← (ACr), EA = (ACxr) + disp
Store Indirect	B000	ST 0,@	1 0 1 1 0 0	(EA) ← (AC0), EA = ((ACxr) + disp)
Load with Sign Extended	BC00	LSEX 0,	1 0 1 1 1 1	(AC0) ← (EA) bit 7 extended
Memory Data Operate Instructions				
AND	A800	AND 0,	15,14,13,12,11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 1 0 1 0 1 0	(AC0) ← (AC0) ∧ (EA)
OR	A400	OR 0,	1 0 1 0 0 1	(AC0) ← (AC0) ∨ (EA)
Add	E000	ADD r, address/disp(xr)	1 1 1 0 r | xr | disp	(ACr) ← (ACr) + (EA); CY, OV
Subtract with Borrow	9000	SUBB 0,	1 0 0 1 0 0	(AC0) ← (AC0) + ~(EA) + CY; CY, OV
Decimal Add	8800	DECA 0,	1 0 0 0 1 0	(AC0) ← $(AC0)_{10}$ + $(EA)_{10}$ + CY; CY, OV

TABLE IV . Microcomputer Instruction Set (continued).

Description	Opcode Base	Source Statement		Instruction Format	Operation
Register Data-Transfer Instructions					
Load Immediate	5000	LI	r,data	0 1 0 1 0 0 | r | data	(ACr) ← data (bit 7 extended)
Register Copy	5C00	RCPY	sr,dr	0 1 0 1 1 1 | dr | sr | 0 0 0 0 0 0	(ACdr) ← (ACsr)
Register Exchange	6C00	RXCH	sr,dr	0 1 1 0 1 1 | dr | sr | 0 0 0 0 0 0	(ACdr) ↔ (ACsr)
Exchange Register and Stack	1C00	XCHRS	r	0 0 0 1 1 1 | r | 0 0 0 0 0 0 0 0	(ACr) ↔ (STK)
Copy Flags into Register	0400	CFR	r	0 0 0 0 0 1 | r | 0 0 0 0 0 0 0 0	(ACr) ← (FR)
Copy Register into Flags	0800	CRF	r	0 0 0 0 1 0 | r | 0 0 0 0 0 0 0 0	(FR) ← (ACr)
Push Register onto Stack	6000	PUSH	r	0 1 1 0 0 0 | r | 0 0 0 0 0 0 0 0	(STK) ← (ACr)
Pull Stack into Register	6400	PULL	r	0 1 1 0 0 1 | r | 0 0 0 0 0 0 0 0	(ACr) ← (STK)
Push Flags onto Stack	0C00	PUSHF		0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0	(STK) ← (FR)
Pull Stack into Flags	1000	PULLF		0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	(FR) ← (STK)
Register Data-Operate Instructions					
Register Add	6800	RADD	sr,dr	0 1 1 0 1 0 | dr | sr | 0 0 0 0 0 0	(ACdr) ← (ACdr) + (ACsr); CY,OV
Register Add with Carry	7400	RADC	sr,dr	0 1 1 1 0 1 | dr | sr | 0 0 0 0 0 0	(ACdr) ← (ACdr) + (ACsr) + CY; CY,OV
Register AND	5400	RAND	sr,dr	0 1 0 1 0 1 | dr | sr | 0 0 0 0 0 0	(ACdr) ← (ACdr) ∧ (ACsr)
Register Exclusive-OR	5800	RXOR	sr,dr	0 1 0 1 1 0 | dr | sr | 0 0 0 0 0 0	(ACdr) ← (ACdr) ▽ (ACsr)
Complement and Add Immediate	7000	CAI	r,data	0 1 1 1 0 0 | r | data	(ACr) ← ~ (ACr) + data (bit 7 extended)
Shift and Rotate Instructions					
Shift Left	2800	SHL	r,n,ℓ	0 0 1 0 1 0 | r | n | ℓ	(ACr) ← (ACr) shifted left n places, w/wo LINK; B
Shift Right	2C00	SHR	r,n,ℓ	0 0 1 0 1 1 | r | n | ℓ	(ACr) ← (ACr) shifted right n places, w/wo LINK; B
Rotate Left	2000	ROL	r,n,ℓ	0 0 1 0 0 0 | r | n | ℓ	(ACr) ← (ACr) rotated left n places, w/wo LINK; B
Rotate Right	2400	ROR	r,n,ℓ	0 0 1 0 0 1 | r | n | ℓ	(ACr) ← (ACr) rotated right n places, w/wo LINK; B
Miscellaneous Instructions					
Halt	0000	HALT		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Halt
Set Flag	3080	SFLG	fc*	0 0 1 1 | fc | 0 0 0 0 0 0 0 0	$(FR_{fc}) \leftarrow 1$
Pulse Flag	3000	PFLG	fc*	0 0 1 1 | fc | 0 0 0 0 0 0 0 0	$(FR_{fc}) \leftarrow 1$, $(FR_{fc}) \leftarrow 0$
No Operation	5C00	NOP		0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0	(PC) ← (PC) + 1

TABLE V Sample Program

Memory Location	Data Value	Instruction			Comments
40	CD09		LD	AC3, S	Load AC3 with start address
41	C90A		LD	AC2, S1	Load AC2 with new start address.
42	5001		LI	AC1, +1	Load immediately AC1 with value of 0001.
43	C300	Loop:	LD	AC0, (AC3)	Load AC0 relative to AC3.
44	D200		ST	AC0, (AC2)	Store AC0 relative to AC2.
45	FC4B		SKNE	AC3, E	Compare AC3 with END address and skip next instruction if not equal.
46	0000		HALT		Stop the program.
47	6B00		RADD	AC1, AC3	Add AC1 to AC3, result to AC3.
48	7A01		AISZ	AC2, +1	Add the value of 0001 to AC2.
49	19F9		JMP	Loop	Jump back to Loop.
4A	----	S:	.WORD	----	Starting address.
4B	----	E:	.WORD	----	END address.
4C	----	S1:	.WORD	----	New starting address.

to move a given data set from its present location to any desired location in memory, provided that the starting of the new location is not one of the present data locations. The program starts by loading accumulators 3(AC3) and 2(AC2) with present and future data starting locations, respectively. The instructions are executed relative to the program counter (PC). Referring to Table IV the hexadecimal code for the load instruction (LD) is C000. The first instruction is using AC3, then r equals 3. The value of r is 2 for the second instruction. The first and the second instructions are using relative addressing, hence xr is 01. The displacement is the distance PC has to travel to get the data which is to be loaded into the accumulators. Hence, the displacement is 9 and 10(A) for instruction 1 and 2 respectively. After that accumulator 1 (AC1) is loaded immediately with the value 1. The next part of the program is an iterative loop which starts by loading AC0 with the contents of effective address relative to AC3. It then stores the data relative to AC2. At each loop a comparison between what is in AC3 and the end address is performed. If AC3 is equal to the end address the program will stop, otherwise the contents of AC3 and AC2 will be increased by the value 0001. It is believed that hexadecimal representation is easily understood if the reader refers to Table IV .

3.2.5 Addressing Consideration

The 16-bits for Pace addresses provides access to 65, 636; $(FFFF)_{16}$; (2^{16}), discrete address locations |13|. The on-board memory (DEBUG, RAM, and PROM sockets) and the keyboard/display

occupy a total address space of 4K. The remaining 60K of address space is available for allocation to off-board memory and/or peripheral devices.

As is shown in Fig. 3.4 the input data for the program will occupy locations 4 through 6 and 40 through FF in the on-board RAM. These locations are part of the base page addresses. The main body of the program exists on a PROM which occupies locations F400 through F7FF. The scratch pad for the program is contained in locations 3, and 100 through 126. The memory locations for the PROM are selected via an address strapping header-5G, shown in Fig. 3.5 |13|. The I/O addresses used for communicating between the program and the hardware interface are addresses 9000 and A000, for the output, and address C000, which is used for the input. These addresses are chosen for convenience of decoding and because they are located in the off-board address space.

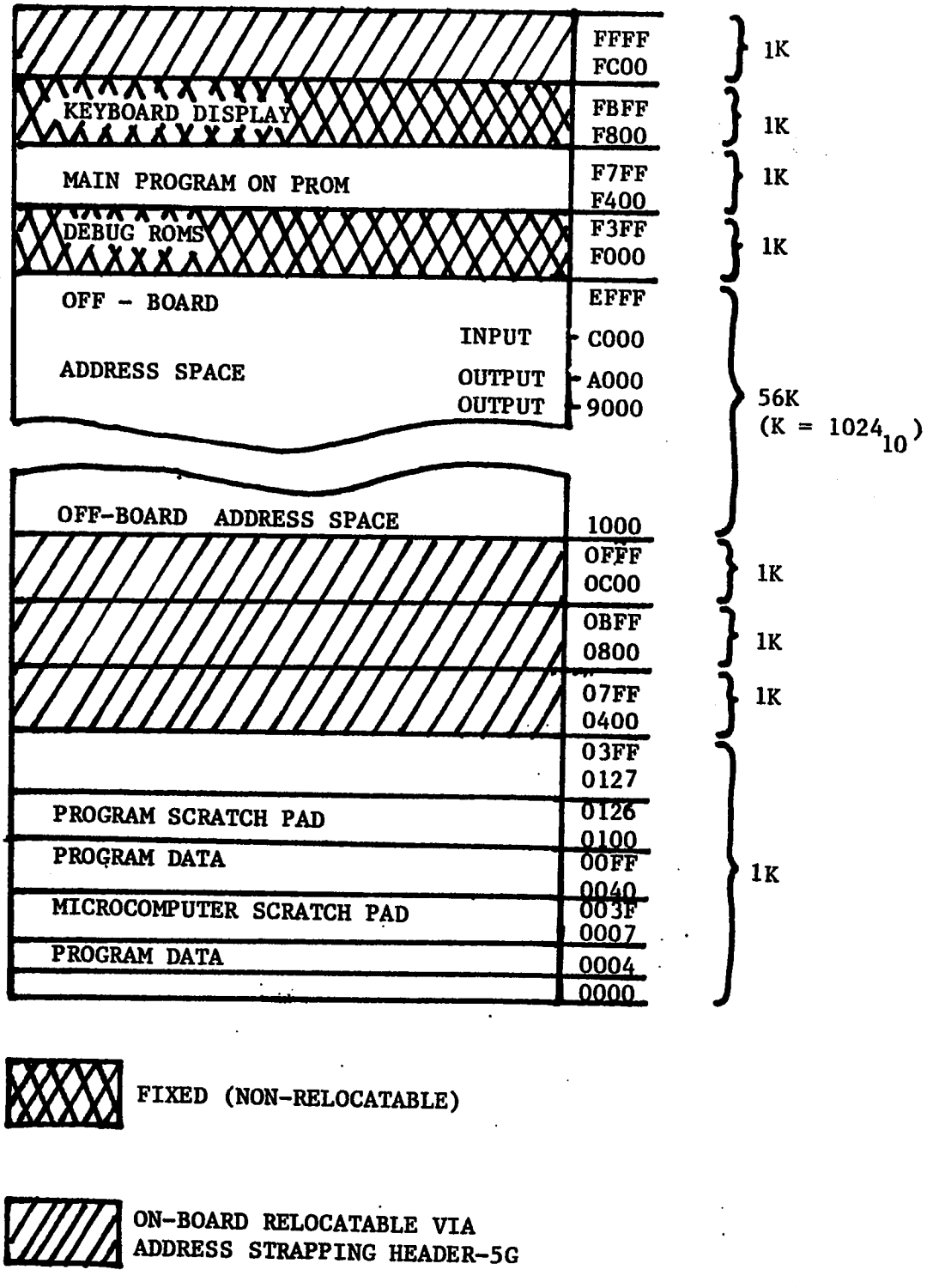


Figure 3.4. LCDS Address Map Used by the Program.

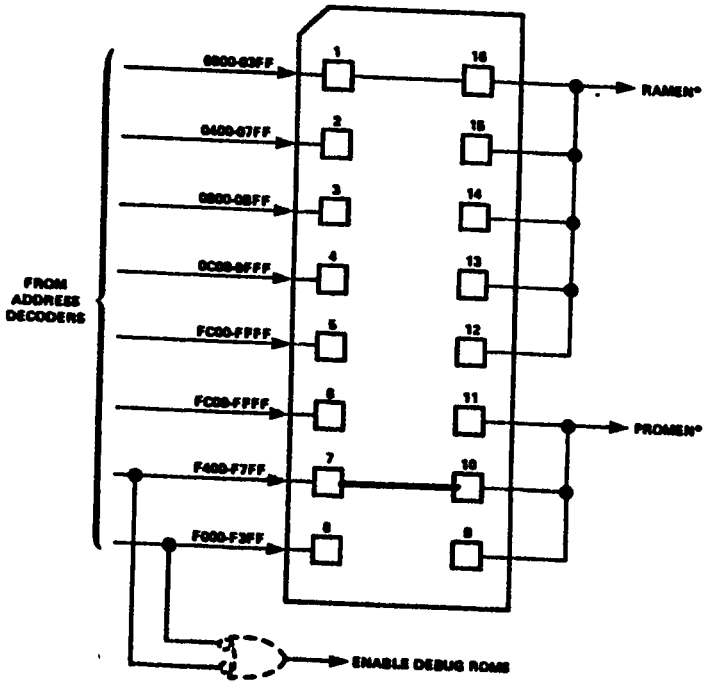


Figure 3.5. Address Strapping Header-5G.

IV. PROGRAMMING ALGORITHM

4.1 OVERVIEW

The microcomputer algorithm is designed to program and verify the PLA in accordance with the instructions provided by the manufacturer (Appendix A). The algorithm programs and subsequently verifies the results for all procedures or for each procedure by itself, as desired. Similarly the algorithm verifies whether a given chip is virgin (unused) for all three device sections, or for each section alone, as required by the user. The algorithm will print out error messages if a verification subroutine encounters an error.

The algorithm uses hexadecimal numbers for input data. This is because the programming is done in machine language for the Low Cost Development System. The principle program limitation is eight output functions, each of which can be a function of as many as sixteen inputs. The quantity of PLA I/O pins is the reason for incorporating this limitation.

About 3.25 seconds are required to burn all fuses of the PLA (1928 fuses). The reader may refer to sections 4.4.2 through 4.4.4 for more details about the time requirements.

The algorithm handles the programming and the verification of the chip according to the flowchart shown in Fig. 4.1. The flowchart shows the major portions of the algorithm. The following sections provide further information with regard to Fig. 4.1.

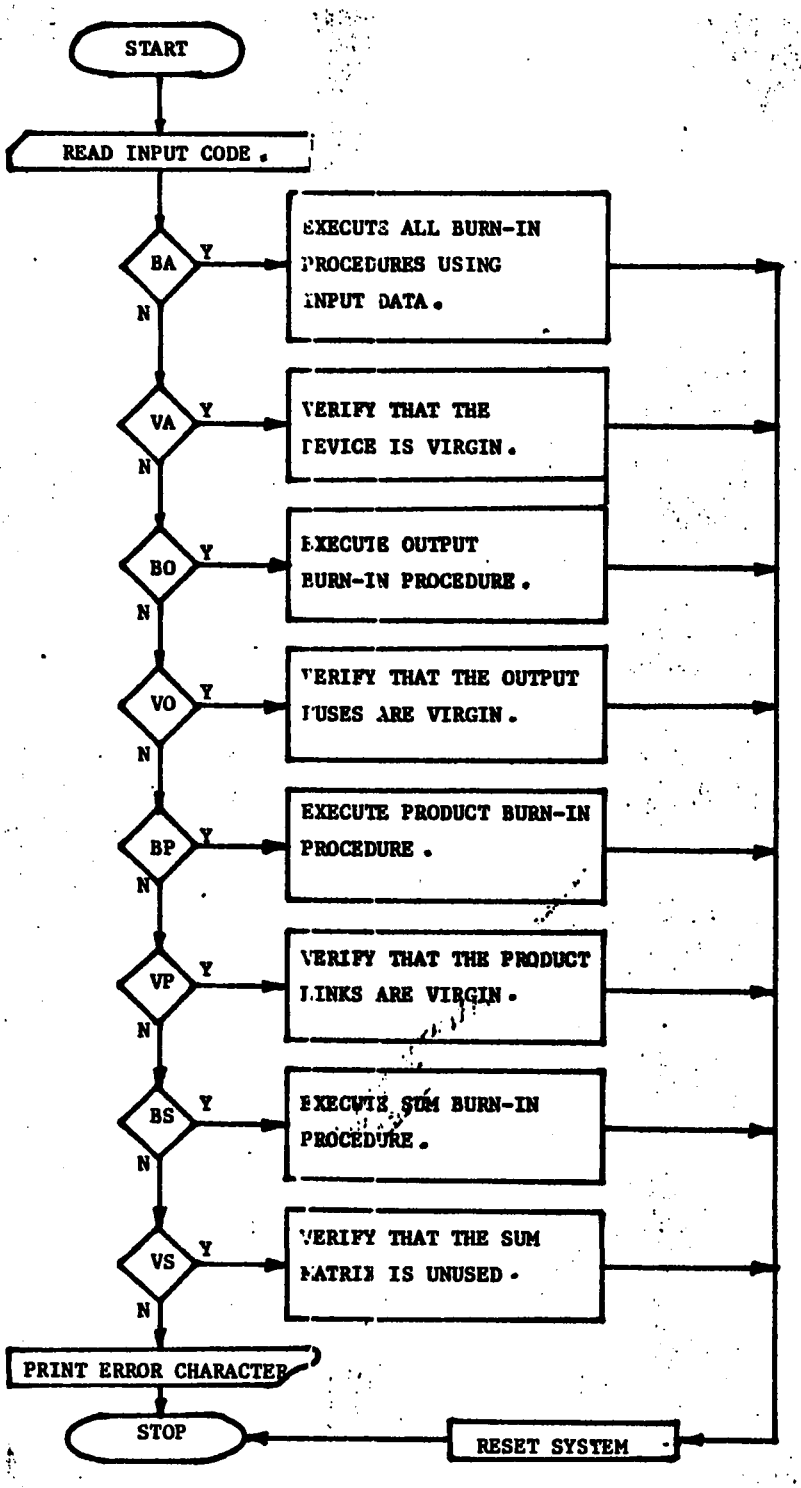


Figure 4.1 Programming Algorithm Flow Chart.

The user can choose the function he desires by punching a two-character code. The functions are either to burn-in fuses or to verify whether the device has been used or not. For example if the user punches "VA"; all fuses of the device will be checked if they are burned or not. While, if he types "BS" the fuses of the sum matrix are going to be burned-in in accordance with the data supplied.

4.2 INPUT FORMAT

4.2.1 General

The input data to the program is as shown in Table VI. The first three input data types supply the system with information about the functions used, the functions to be inverted, and the number of AND gates used, respectively. The rest of the data provide information about each AND gate used. For these AND gates, information about their number, the state of their input variables, and the output functions connected to the gate is provided. The input data uses binary number representations. For example if outputs F_0 and F_7 are to be used, the "FU" vector should contain a 1 in both its zero and seventh bit.

4.2.2 Generating input format from a truth table

The format of the input will be described by means of an example. Consider the programming of the following functions:

TABLE VI . Input Format for the Programming Algorithm. 40

Data input number	Memory location	Description
1	0004	FU = output functions to be used

Bit	7	6	5	4	3	2	1	0
	f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0

$Bit_i = 1$ respective function f_i is used

$(0 \leq i \leq 7)$

2	0005	FI = output functions to be inverted

Bit	8	7	6	5	4	3	2	1	0
	f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0	M

M = 0 no functions to be inverted

$Bit_i = 1$ respective function f_{i-1} is to be inverted. $(1 \leq i \leq 8)$

3	0006	NA = number of AND gates to be used

$(1 \leq NA \leq 30)_{16}$

4	0040	I = number of the AND gate to be programmed

$(0 \leq I \leq 2\ F)_{16}$

5	0041	Product term representing the AND gate used

Bit	15 - - - -	4	3	2	1	0
	I_{15}	I_4	I_3	I_2	I_1	I_0

$Bit_i = 0$ respective variable I_i is to be inverted $(0 \leq i \leq 15)$

TABLE VI . Input Format for the Programming Algorithm (Cont'd).

Data input number	Memory location	Description
6	0042	"don't care" variables in the minterm Bit: 15 ... 3 2 1 0 I_{15} ..... I_3 I_2 I_1 I_0 Bit i = 1 respective variable I_i is to be considered as a"don't care". $(0 \leq i \leq 15)$
7	0043	Output functions connected to the AND gate Bit: 7 ... 2 1 0 f_7 ..... f_2 f_1 f_0 Bit i = 1 respective function is connected to the AND gate. $(0 \leq i \leq 7)$
8	0044-00FF	Data numbers 4 through 7 are repeated for other AND gates to be programmed

$$F_0 (A, B, C, D) = \Sigma (1, 3, 5, 7, 8, 9, 12, 13)$$

$$F_3 (A, B, C, D) = \overline{\Sigma (6, 7, 8, 9, 11, 12, 13, 14, 15)}$$

$$F_7 (A, B, C, D) = \Sigma (2, 8, 10, 11, 12, 15)$$

The functions are shown in decimal representation |23|. In this example, there are three functions to be used (F_0, F_3, F_7). Hence, bits 0, 3, 7 are set to 1 in memory location 4. Function F_3 is to be inverted; implying bits 0 and 4 are set to 1 in location 5. The total number of AND gates is 14, excluding a recount of those repeated. The AND gates of the PLA must be determined by the user. Any arbitrary set of AND gates can be chosen. This selection need not to be in a consecutive sequence. In this example the gate numbers $(10 - 1D)_{16}$ are selected. Since only four variables are used, the rest of the variables (12) are considered as"don't care". Any set of input variables can be selected. Here variables I_4 to I_7 were used. Hence, the data for the"don't care"variables is FFOF. The data for the states of the input variables for the minterm specified as 8(selected to be gate number 14) is 80 because variables I_4 to I_7 are selected. Functions F_0, F_3 and F_7 are connected to gate 14; implying bits 0, 3 and 7 are set high in the data provided for the functions connected to gate 14.

The format of the input, as loaded to the microcomputer via the TTY, is shown in Table VII. The first two characters of each line (AM) represent the loading instruction to the microcomputer via the TTY. Data loading will start at the location typed immediately after these characters and continue for the successive locations.

TABLE VII Input Data Through TTY .

AM04, 19, 11, E

AM40, 10, 10, FFOF, 1, 11, 30, FFOF, 1, 12, 50, FFOF, 1

AM4C, 13, 70, FFOF, 9, 14, 80, FFOF, 19, 15, 90, FFOF, 9

AM58, 16, CO, FFOF, 89, 17, DO, FFOF, 9, 18, 60, FFOF, 8

AM64, 19, BO, FFOF, 81, 1A, EO, FFOF, 8, 1B, FO, FFOF, 8

AM70, 1C, 20, FFOF, 80, 1D, AO, FFOF, 80

4.2.3 Generating the Input Format from the Output of the Minimization Program

The output format of the minimization program is shown in Table VIII. Definitions appear in Table IX |10|. An example will now be presented to explain this format. The prime implicants covering functions 1 and 2 (Table VIII) are indicated as (4,8), (6,4). The first number of each set provides attributes of each input variable used. The second number represents whether an input variable is a "don't care"; i.e. does not appear in a prime implicant. Variables not used by the functions have zeros in their respective bits in both numbers. For example the first PI typed can be obtained from the printed set of (4,8) as follows:

$$
\begin{array}{rcccc}
(4,8)_{10} = & A & B & C & D \\
 & 0 & 1 & 0 & 0_2 \\
 & 1 & 0 & 0 & 0_2 \\
\hline
 & - & B & \overline{C} & \overline{D}
\end{array}
$$

Since F_1 and F_2 are functions of A, B, C, D only, the other bits in the numbers appear as zeros.

The program was modified to get an output format suitable for the burn-in program. The flow chart for the modification is shown in Fig. 4.2. The steps performed to obtain the new format are:

1. The modified program stores output data of the original program, for the AND gates, in any array MST (48,10) following the format below:

TABLE VIII Output Format of Minimization Program .

* * * * * PRIME IMPLICANTS COVERING FUNCTION-1
(6, 6)
(1, 3)
* * * * PI-S COVERING PROD FUNS. F-1, 2,
(4, 8)
(6, 4)

TABLE IX Definitions for the Output Format of the Minimization Program.

Element	Description
PI	Prime implicants
PROD FUNC i	Output functions to be minimized with their respective numbers. $1 \leq i \leq 8$
(A,B)	A is a binary representation of input variables if the input variable respective bit is 0 it means that the input variable is complemented. Otherwise it is uncomplemented. B is a binary representation of input variables. If the input variable respective bit is 1 it indicates that the input variable is a"don't care".

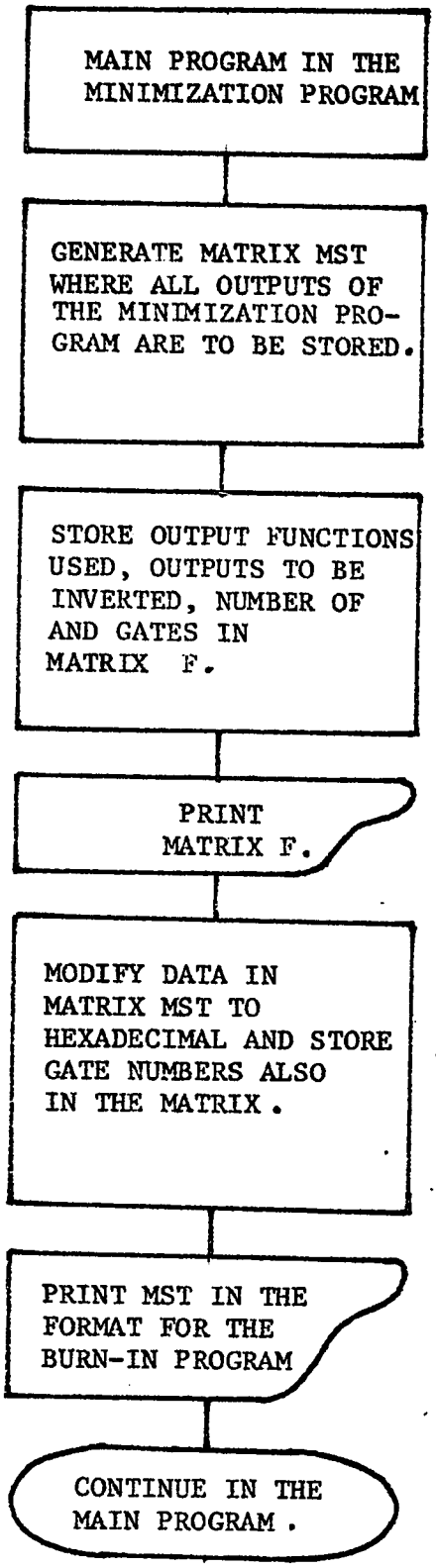


Figure 4.2. Flow Chart of Modifications Added to the Main Program.

1) (I,1) contains the decimal product term representation of the AND gate.

2) (I,2) contains the decimal"don't care" representation in the AND gate.

3) (I,3) to (I,10) indicate output functions utilizing the AND gate.

4) $1 \leq I \leq 48$; i.e. steps 1 through 3 are repeated for as many as 48 AND gates.

2. 1) The data in a vector F (8 x 1) indicates which are the inverted functions. This information is converted to binary representation. Since the data provided in vector F is only temporarily needed, certain locations of the array are later used for other purposes. For example, the binary representation of the functions requiring inversion is stored in location 2 of the array. This can be done since only one word is needed for the binary representation. Then, location 2 is increased by one, if it is greater than zero, in order to conform to the format of the microcomputer FI vector.

2) The total number of AND gates (LNPI) is stored in location 3 of vector F.

3) The functions used are specified by a single decimal number. The respective function bit in a binary equivalent of the decimal number is set high if the function is used. The decimal number is stored in location 1 of array F.

4) Elements F(1) through F(3) are printed in hexadecimal numbers with the known microcomputer starting memory locations.

3. 1) The data in elements (I,3) through (I,10) is represented in binary by one number. This number is stored in (I,4), because the information provided in elements (I,3) to (I,10) is no longer needed.

2) The data in element (I,2) is modified by setting the bits representing unused input variables to one. The result is stored in location (I,3).

3) Element (I,1) is stored in (I,2) to suit the input format of the burn-in program.

4) The AND gate number is stored in (I,1).

5) Elements of the set {(I,J)}, where J = 1 to 4 and I = 1 to LNPI, are printed as hexadecimal numbers. The elements represent required information about each AND gate. Information for each three AND gates is printed on one line, along with the first loading location of the information in the microcomputer memory area.

The example output of the modified program is shown in Table X. Referring to the example of the previous section, the input variables are I_0 to I_3 and the AND gates are 00 to D_{16}. Functions 0, 1, 2 correspond to functions F_0, F_3, F_7, respectively. This is because the user does not have the specification flexibility for the minimization program that exists for the microcomputer

TABLE X. Output Format of the Modified Minimization Program .

AM04, 0007, 0003, 000E

AM40, 0000, 0001, FFFO, 0001, 0001, 0003, FFFO, 0001, 0002, 0005, FFFO, 0001

AM4C, 0003, 0007, FFFO, 0003, 0004, 0008, FFFO, 0007, 0005, 0009, FFFO, 0003

AM58, 0006, 000C, FFFO, 0007, 0007, 000D, FFFO, 0003, 0008, 0006, FFFO, 0002

AM64, 0009, 000B, FFFO, 0005, 000A, 000E, FFFO, 0002, 000B, 000F, FFFO, 0002

AM70, 000C, 0002, FFFO, 0004, 000D, 000A, FFFO, 0004

program. The computer listing for the modification is shown in Appendix E.

4.3 I/O SIGNALS FOR THE HARDWARE INTERFACE

As described before, two addresses are used to transmit data to the hardware; namely 9000, and A000. The other address C000 is used to verify the burning procedures of the PLA. The first two are output signals and the third is an input signal with respect to the microcomputer. The output addresses are used to provide 29 bits of code to the interface circuit. These 29 bits are divided into eight vectors. The vectors are defined in Table XI. The Verify vector uses only one bit. This bit is located in position zero of the word located at C000. Its purpose is to sense the state of a given PLA output under certain defined conditions. The algorithm reads this bit and checks whether it is correct in accordance with pre-programmed information.

4.4 SOFTWARE DESCRIPTION

4.4.1 Overview

The software is divided into three major sectors, as follows:

1. Output,
2. Product,
3. Sum.

Each portion is further divided into two subprograms:

TABLE XI Definition of Input/Output Address Vectors.

ADDRESS	VECTOR NAME	BIT	NAME	DESCRIPTION
A000	PLA I/O	0-3	P_0-P_3	Specifies which input or output is to be effected in the burn-in step. It is represented in a binary code and therefore requires decoding.
A000	Address	8-13	A_0-A_5	Used to address inputs or outputs by supplying binary values to respective inputs or outputs.
C000	Verify	0	V	This bit is used to transmit the result of sensing a given output to the microcomputer.
9000	Supply	0	B_{CCP}	When high V_{CC} will be at V_{CCP} (+5V).
9000	Voltage	1	B_{CCS}	When high V_{CC} will be at V_{CCS} (+8.5V).
9000	(V_{CC})	2	B_{CCL}	When high V_{CC} will be at V_{CCL} (0 V). (When all low V_{CC} will be Open.)
9000	FIELD	3	B_{FEH}	When high F_E will be at V_{FEH} (17V).
9000	ENABLE (FE)	4	B_{FEL}	When high F_E will be at V_{FEL} (0V). (When both are low F_E will be open.)
9000	CHIP	5	B_{IXCE}	When high $\overline{CE}$ will be at V_{IX} (10V).
9000	ENABLE ($\overline{CE}$)	11	$B_{\overline{CE}}$	When high, provided V_{IXCE} and B_{ICE} are low, $\overline{CE}$ will be at 0V. Or at 5V if I_{CE} is high. When B_{IXCE} and $B_{\overline{CE}}$ are low, $\overline{CE}$ is open.
9000		14	B_{ICE}	

TABLE XI Definition of Input/Output Address Vectors (Cont'd).

ADDRESS	VECTOR NAME	BIT	NAME	DESCRIPTION
9000	INPUT	6	B_{IH}	When high all inputs are at V_{IH} (5V).
9000	Variables	12	B_{IX}	When high, provided A and $\overline{P}$ are low, all inputs are at V_{IX} (+10V).
A000	(I)	4	A	When high, inputs I_0 thru I_5 will be at voltage levels specified by ddress vector (0, or 5V).
A000		6	$\overline{P}$	When high, the given input variable
A000		14	A_I	specified by the PLA I/O vector will assume the value of A_I (0 or 5V). (When bits 6, 12, 4 and 6 are low inputs are open.)
9000	OUTPUT	7	B_{OPH}	When high, the output function specified by PLA I/O is set to V_{OPH} (+17V).
9000	Variables	8	B_{OPL}	When high, all output functions are set to V_{OPL} (0V).
9000	(F)	9	B_{OPF}	When high, the output function specified by PLA I/O is set to V_{OPF} (+10V).
9000		10	B_{sense}	When high, the output function specified by PLA I/O is sensed.
9000		13	$\overline{B_{sen7}}$	When high, the output function 7 is sensed.
A000		5	A_F	When high, the output functions F_0 through F_5 assume the voltage levels in the address vector. (When all are low, the voltage outputs are open.)

1. Procedure burn-in and verification,

2. Virgin device verification for a given procedure.

The programs include the following features:

1. The average time to execute an instruction is approximately 12 μsec. Hence, the minimum pulse sequence delay is about 24 μsec, which fits the PLA specifications (Appendix A) for a minimum delay between steps (t_D) of 10 μsec.

2. Delay routines to achieve a programming pulse width (t_p) in excess of 100 μsec (Appendix A).

3. Error messages, which are printed via TTY, if any error is encountered in verifying the PLA. This part will be discussed in section 4.4.5.

4. The user can choose to burn-in any sector or a complete burn-in. Also, verification can be done separately. The user can apply one of the symbols shown in Table XII to the TTY. Or he can write a program, in any locations between 0127 and 03FF, and supply the code for the particular part in the specified location, as described in Table XIII. This alternative is used when the user wants to link the burn-in program to his own program. The software assembly language listing, its machine code, and memory locations of the burn-in program are shown in Appendix D.

4.4.2 Program "Output"

TABLE XII Input Symbols and Their Codes.

CODE	SYMBOL	DESCRIPTION
4241	BA	Execute all burn-in procedures and verify what has been burned using given data.
5641	VA	Verify that the device is virgin.
424F	BO	Execute the output burn-in and verify that output fuses indicated in the data are burned.
564F	VO	Verify that output fuses are still intact.
4250	BP	Execute the product burn-in and verify that product fuses specified in the data are burned.
5650	VP	Verify that product matrix fuses are intact.
4253	BS	Execute the sum burn-in and verify that burn-in is successful according to input data.
5653	VS	Verify that sum fuses are virgin.

TABLE XIII Program Used as an Alternative to the Symbols of Table XII

HEXADECIMAL CODE	ASSEMBLY LANGUAGE		
C505	LD	AC1, CODE	
9505	JSR	@ TEST	
0000	HALT		
5020	LI	AC0,20	
0000	HALT		
(CODE)	CODE:.WORD	----	; SYMBOL
F401	TEST:.WORD	0F401	

4.4.2.1 Burn-in "OUT" program

The OUT program flowchart is shown in Fig. 4.3. This program starts by examining memory location 5 to find if there are any outputs to be inverted. If this happens to be the case, it initializes the PLA pins in accordance with Fig. 4.4. Afterwards, an iterative loop will take place. The function of this loop is to burn the exclusive OR gates buses of the F_J's which require inversion. The F_J's requiring inversion are supplied in memory location 0005, by identifying the index J. This burn-in is performed by applying a high voltage to the appropriate F_J pin. After the fuse of the F_J pin, needing inversion has been burned, verification will start. Verification is done to verify whether the burn-in has been accomplished or not. Verification starts by resetting the burn-in signal. Then signals are set to initialize verification. The appropriate F_J pin is sensed and the result is transmitted back to the microcomputer. The result of verification should be "high", which means that the corresponding exclusive OR gate has been fused. Should an error be encountered, it will be stored in memory with its F_J number. When all F_J pins, needing inversion are fused and verified, the microcomputer will send reset signals to the PLA. If an error has occurred (memory location 10A), error messages will be printed out. The reader may refer to section 4.4.5 for more information about error messages. These error messages will tell whether the fuse of the output, needing inversion, has not been burned.

An example function of the microcomputer output code is shown in Fig. 4.5. It is used to burn the fuse of output function 7. The

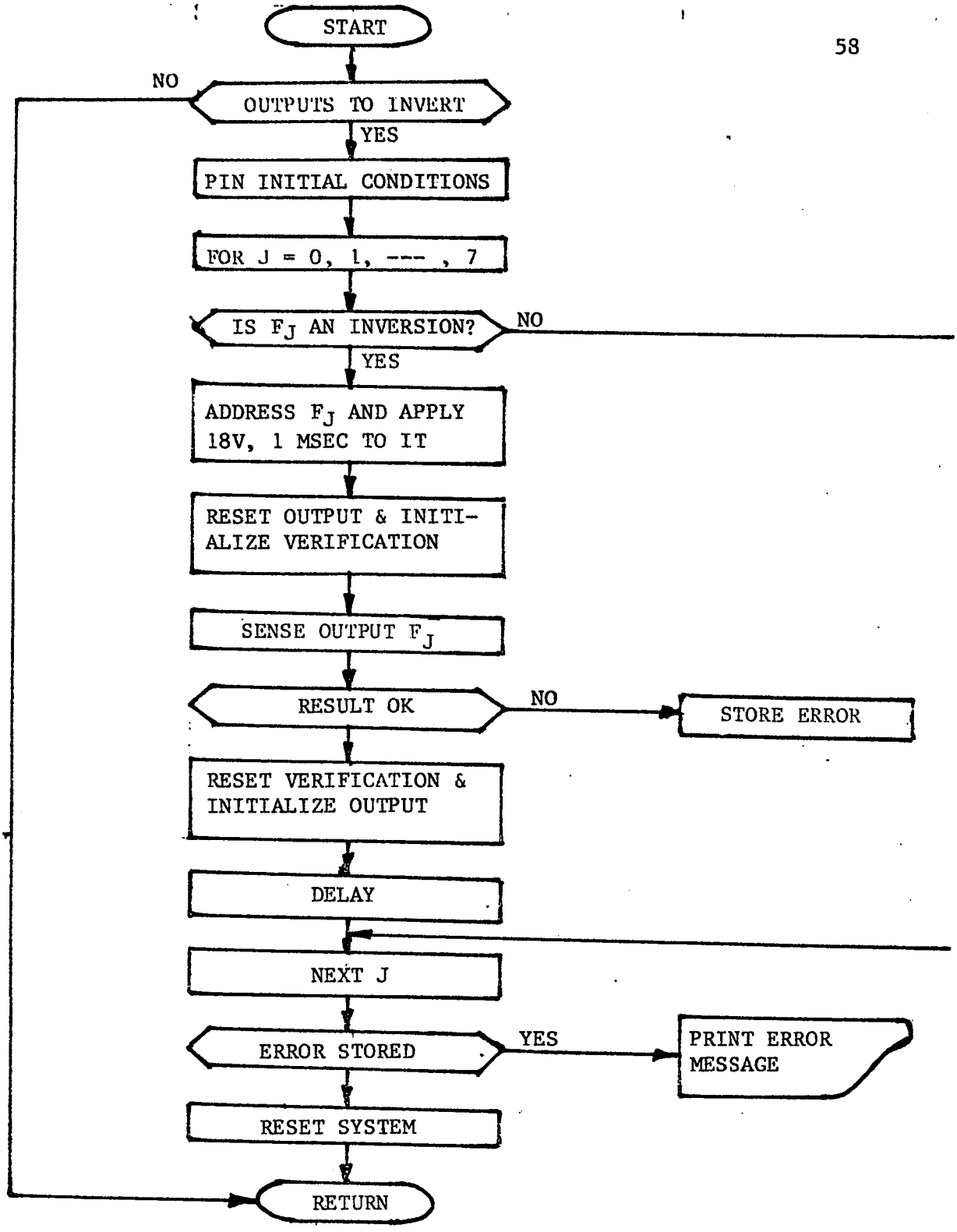


Figure 4.3 Output Burn-in and Its Verification Flow Chart.

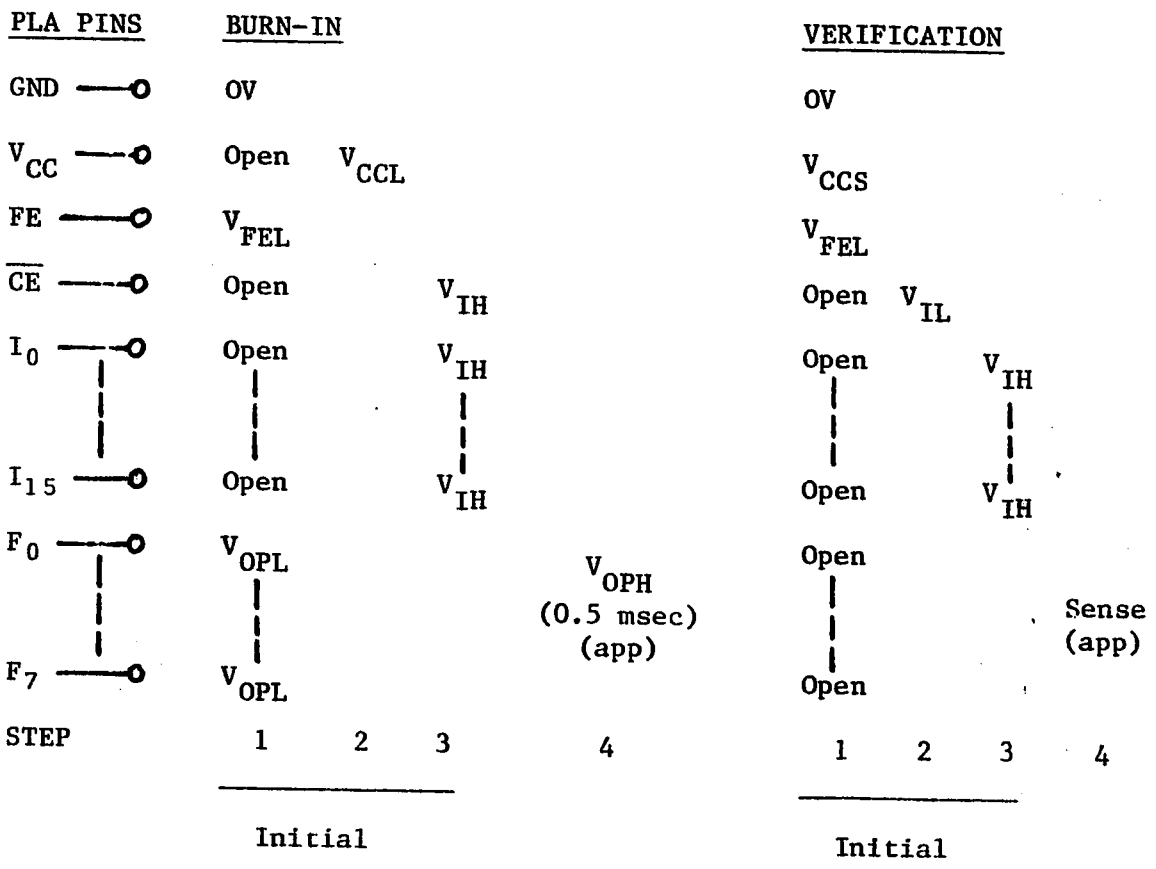


Figure 4.4 PLA Pin Input Sequence for Output Burn-in and its Verification.

STEP	DATA																DATA
	9000	B_{ICE}	$B_{\overline{sen7}}$	B_{IX}	$B_{\overline{CE}}$	B_{sense}	B_{OPF}	B_{OPL}	B_{OPH}	B_{TH}	B_{IXCE}	B_{FEL}	B_{FEH}	B_{CCL}	B_{CCS}	B_{CCP}	
1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	
2	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	A000
3	0	1	0	0	1	0	0	1	0	1	0	1	0	1	0	0	$A_I A_5 A_4 A_3 A_2 A_1 A_0$ $\overline{P}$ A_F A $P_3 P_2 P_1 P_0$
4																	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
5	0	1	0	0	1	0	0	1	1	1	0	1	0	1	0	0	
DELAY Programmed in microcomputer for 0.5 msec.																	
6	0	1	0	0	1	0	0	1	0	1	0	1	0	1	0	0	
7	0	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	
8	0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0	
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
10	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	
11	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	
12	0	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	C000 V
13			Sense F														0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
14	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	
15	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
17	0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0	
18	0	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	
19	0	1	0	0	1	0	0	1	0	1	0	1	0	1	0	0	
DELAY Programmed in microcomputer for 0.5 msec.																	
20	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	
21	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
23																	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4.5 Code to Burn Output 7 and its Verification

procedure starts by initializing the output sending codes (steps 1, 2, and 3). This entails a time duration of about 24 μsec for each step. Afterwards output 7 is addressed using the PLA I/O vector in step 4. To burn the fuse connected to the output, a pulse of 17 volts for a duration a 0.5 msec is sent in steps 5 and 6. Reset steps 7 and 8 are sent to transfer the PLA from burn-in to verification. Steps 9, 10 and 11 are used to initialize verification in accordance with Fig. 4.4. Step 12 is sent to sense output 7. The result of the sensing procedure is sent back to the micrcomputer via vector V. Steps 14, 15, 16, 17, 18 and 19 are sent to reset verification and to become ready for burn-in of other functions. A delay routing of about 750 μsec is required between a burn-in fuses. The microcomputer accomplishes this delay via software. Since no more outputs to be burned steps 20, 21, 22, 23 are performed to remove the output signal and reset the system. The time function for the output burn-in is:

$$F(J) = 175 \times 10^{-6} + 1.5 \times 10^{-3}(J) \quad \text{sec.}$$

Where: J is the number of functions to be inverted.

4.4.2.2 Verify "OUTV"

This program's flowchart is shown in Fig. 4.6. It starts by initializing the PLA pins as shown in Fig. 4.4. Then an iterative loop will take place. The function of this loop is to verify that all PLA output pins are unfused (virgin). The verification is performed by sending a code for output to be verified and then sensing the result. If one of the outputs is fused, its

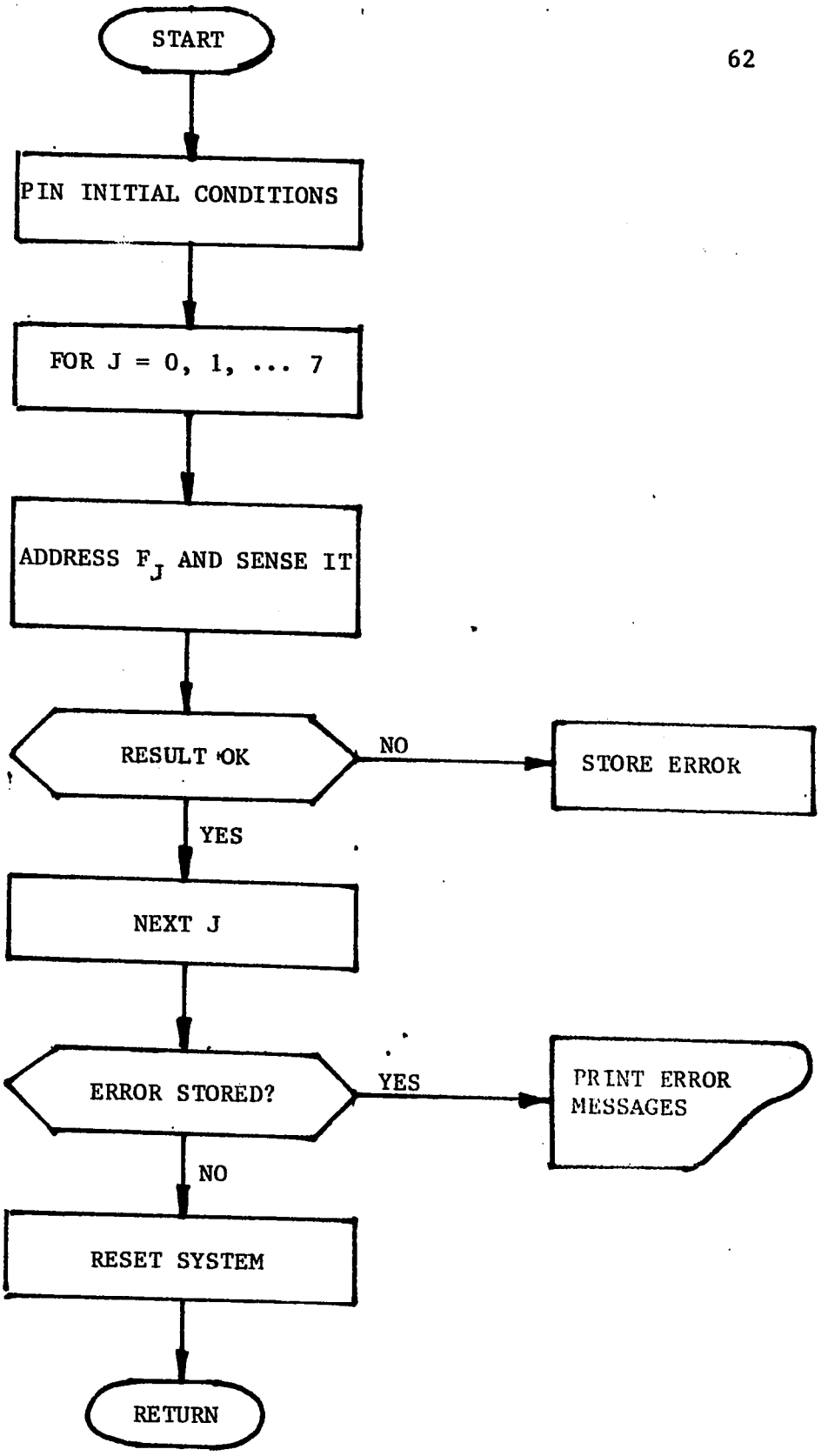


Figure 4.6 Output Verification (OUTV) Flow Chart.

will be stored to be printed later as an error message. After all outputs have been verified, error messages, if any, are printed. The microcomputer will then send reset signals.

The code shown in Fig. 4.7 illustrates the verification for output F_0. The program starts by performing steps 1, 2 and 3 to initialize PLA pins in accordance with Fig. 4.4. The appropriate output pin is addressed as in step 4. In this case F_0 is addressed. Then step 5 is performed to execute the sensing function. The result is transferred back to the microcomputer. Step 6 resets the PLA pins back to initial conditions. Other outputs are sensed by means of iterating steps 4, 5 and 6. When all outputs are sensed steps 7, 8, 9, and 10 are performed to reset the system. The time required for the verification of outputs is about 1150 μsec.

4.4.3 Program "Product"

4.4.3.1 Burn-in routine "PROD"

The formation of the product is more complicated than the preceding. Each of the 16 inputs will have one of three conditions for a product term (p-term): 1. appears as an uncomplemented literal, 2. appears as a complemented literal, or 3. does not appear. Prior to burn-in, two fuses connect each input to the product AND gate. One fuse provides the path for an uncomplemented literal and the other is used for a complemented literal. Both fuses are burned if its corresponding variable does not appear in the product. As shown in the flowchart of Fig. 4.8 the procedure

STEP	DATA																DATA
	9000																
		B_{ICE}	$\overline{F_{sen7}}$	B_{IX}	$\overline{B_{CE}}$	B_{sense}	B_{OPF}	B_{OPL}	B_{OPH}	B_{IH}	B_{IXCE}	B_{FEL}	B_{FEH}	B_{CCL}	B_{CCS}	B_{CCP}	
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
2	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	A000
3	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	$A_I A_5 A_4 A_3 A_2 A_1 A_0 \ \overline{P} \ A_F$ A $P_3 P_2 P_1 P_0$
4																	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	0	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	C000
			Sense F_0														0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	
			Sense other outputs														
7	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	
8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
10																	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4.7 Code to Verify Virgin Status of Output F_0.

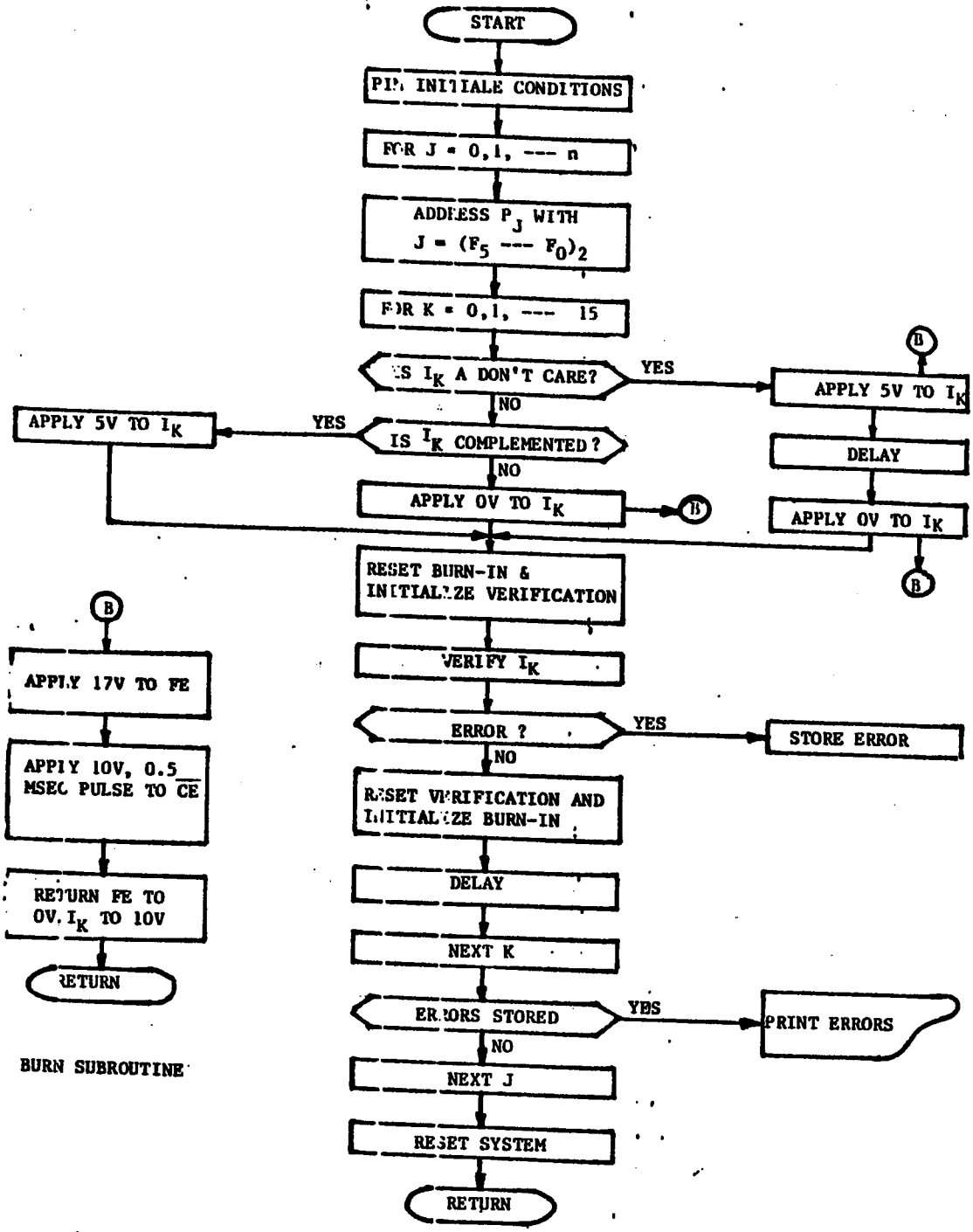


Figure 4.8 Flow Chart of the Product Burn-in and Its Verification (PROD).

starts by initializing the PLA pins as illustrated in Fig. 4.9. Afterwards, the p-term is addressed by providing a binary equivalent to the PLA pins $F_5 \ldots F_0$. For example P_6 requires the pin voltage (0, 0, 0, 5, 5, 0). The inner loop, index K, performs the variable I_K burn-in. If the I_K entry is "0", then a complemented literal is to appear within P_J. Observe that burn-in needs to occur twice if I_K does not appear in the product. After completion of the burn-in for each literal, verification will take place. Verification starts by resetting the PLA pin signals and then initializing for the verification, as shown in Fig. 5.9. If an error is encountered, it will be stored with its respective variable identifying number (J). After all literals of a given P-term are fused, error messages, if any, will be printed. When all P-terms are fused, a reset signal will be sent to the PLA.

To illustrate the codes sent to the hardware in the PROD program, the steps in Fig. 4.10 are used to burn-in input 3, for P-term 16, as a "don't care". The microcomputer software starts by initializing the system as shown in steps 1, 2 and 3. Then the P-term used is addressed by sending the codes of steps 4 and 5. Afterwards, step 6 sends the code to address the input to be bunred, in this case input number 3 is set to high logic level. The burn-in starts by setting FE to V_{FEH} (17.5V); (step 8), and pulsing $\overline{CE}$ to V_{IX}(10.5V) for 0.5 msec (step 9). Steps 10 and 11 will return $\overline{CE}$ to 5V and FE to V_{FEL} (zero volts). According to the manufacturer specifications as illustrated in Appendix A |12|, a delay should take place between each burn-in. Afterwards input 3 is set to the low logic level and its fuse is burned by performing steps 12, 13, 14

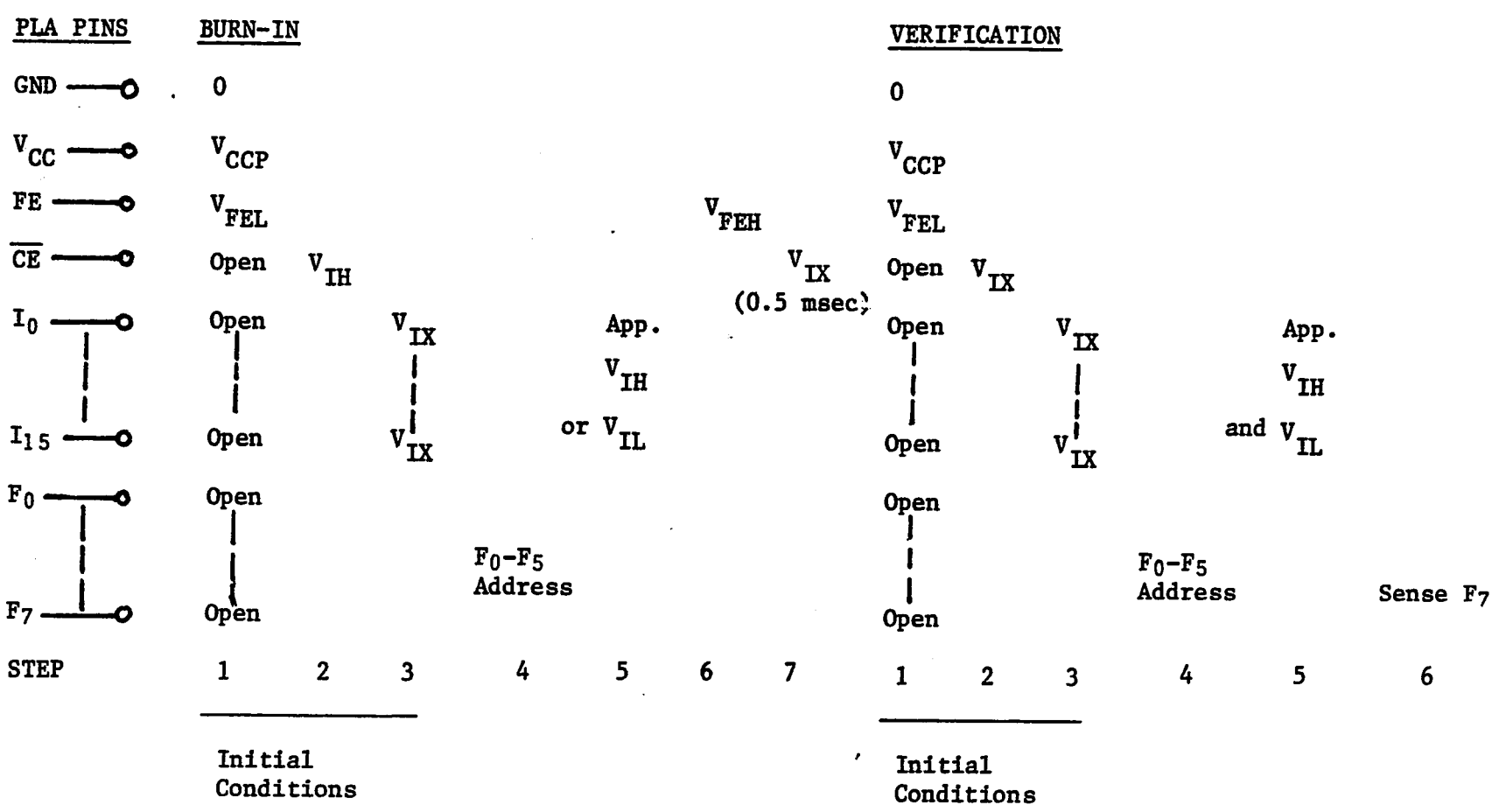


Figure 4.9 PLA Pin Input Sequence for Product Burn-in and its Verification.

STEP	DATA																DATA									
	9000																									
		B_{ICE}	$\overline{B_{sen7}}$	$\overline{B_{IX}}$	$\overline{B_{CE}}$	B_{sense}	B_{OPF}	B_{OPL}	B_{OPH}	B_{IH}	B_{IXCE}	B_{FEL}	B_{FEH}	B_{CCL}	B_{CCS}	B_{CCP}										
1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1										
2	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	A000									
3	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1	A_I	A_5	A_4	A_3	A_2	A_1	A_0	$\overline{P}$	A_F	A	P_3	P_2	P_1	P_0
4																	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5																	0	0	0	1	0	0	0	0	0	1	0	0	0	0
6																	0	1	0	1	0	0	0	0	0	1	0	0	0	1
7																	0	1	0	1	0	0	0	0	0	1	0	0	0	1
8	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1										
9	0	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1										
10	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1										
11	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1										
DELAY						Programmed delay of 750 μsec.																				
12																	0	0	0	1	0	0	0	0	0	1	1	0	0	0	1	1
13	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1										
14																										
14	0	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1										
DELAY						Programmed delay of 0.5 msec.																				
15	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1										
16	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1										
17																	0	1	0	1	0	0	0	0	0	1	1	0	0	0	1	1
18	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0	1										
19	0	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1	C000									
20						Sense											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	V 1
																	A000									
21																	0	0	0	1	0	0	0	0	0	1	1	0	0	0	1	1
																	C000									
22						Sense											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
23	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0	1										
24	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1	A000									
25																	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1
DELAY						Programmed delay of 750 μsec.																				
26						(After finishing all inputs)											0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
						(After finishing all P-terms)																				
27	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1										
28	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1										
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000									
30																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.10 Steps Used to Burn-in Input 3 for P-term 16 as a "don't care" and its Verification.

and 15, which are similar to steps 8, 9, 10, and 11. At step 17, the verification starts by setting input 3 to a high logic level and initializing the verification procedure. This is accomplished within steps 18 and 19. The microcomputer will receive the output of the sensing function at address C000 (bit 0) and will store it. At steps 21, input 3 is set to the low logic level and output 7 is sensed for verification. Steps 23 and 24 send codes to reset verification and initialize burn-in. When verification of a given input is finished, the address signal for it will be removed. A delay will now transpire when all inputs are verified for a given P-term, the signal addressing the P-term will be removed (step 27). When all P-terms have been verified, the reset codes will be sent to the hardware. Steps 28, 29, and 30 are performed for the reset codes.

The time function for the PROD program is:

$$F(J) = 250 \times 10^{-6} \text{ 1 sec} + 1.5 \times 10^{-3}(J) \quad \text{sec.}$$

Where J is the number of fuses to be burned.

4.4.3.2 Verification "PRODV"

The program PRODV flowchart is shown in Fig. 4.11 |12|. The PLA pins are first initialized as shown in Fig. 4.9 |12| and the P-term to be verified, is afterwards addressed. Verification is performed by sensing output number 7 for all input variables. For each input, output 7 is sensed twice; once when the input is high, and the other time when input is low. If the result is not zero this indicates

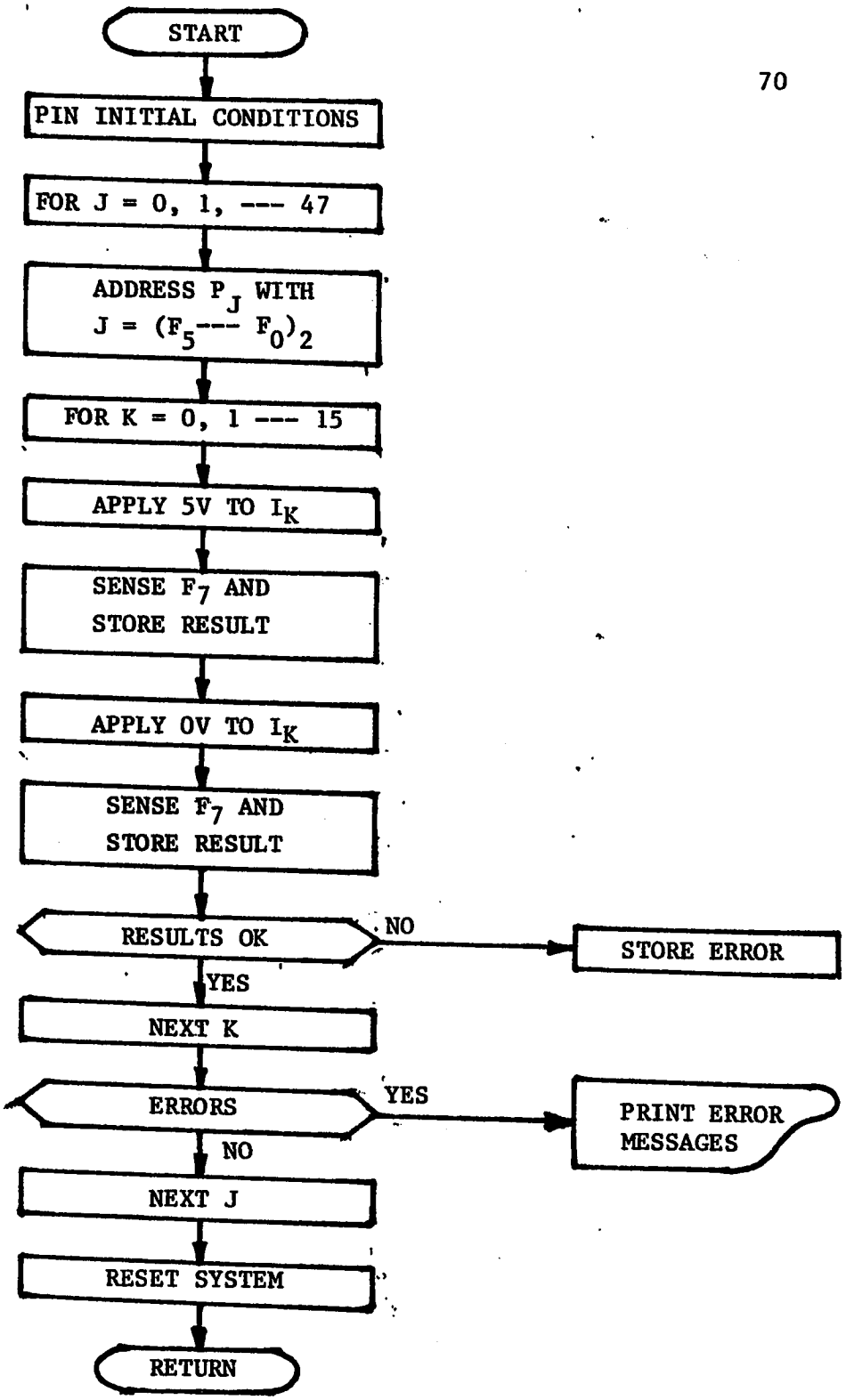


Figure 4.11 Verification of Product Terms Flow Chart (PRODV).

the input is not virgin, and an error message will be stored. The error message will contain the input number and its state. After verification all inputs, of a particular P-term, error messages will be printed. When completing the verification of all P-terms, the reset signal will be sent.

As an example to illustrate the product verification, the coding in Fig. 4.12 verifies input I_5 of P-term 20. The microcomputer starts by initializing the PLA pins for PRODV. Steps 1, 2 and 3 show the initializing codes. Then, P-term 20 is addressed by sending codes shown in steps 4 and 5. Input I_5 is addressed by during steps 6, 7 and 8. In step 8, the input I_5 is set to a high logic value. Sense code 9 is now sent to the hardware to sense output 7 and the result is transmitted back to minicomputer via address C000. The sensing is done twice to fully identify the state of an input. In step 11 I_5 was set to low logic. Step 13 is used to reset sensing function. Step 14 resets the input address. When all inputs have been verified, the code of step 15 is sent to reset the address of the product terms. When all P-terms have been verified, the system is reset by steps 17, 18, 19 and 20.

The time required to verify the AND matrix is about 115 msec.

4.4.4 Program "Sum"

4.4.4.1 Burn-in "SUM"

The sum burn-in procedure is similar to that for the product. The major distinction is that the inputs $I_5, \ldots, I_0$ form the binary

STEP	DATA																DATA
	9000																
		B_{ICE} / B_{sen7}		B_{IX} / B_{CE}		B_{sense}	B_{OPF}	B_{OPL}	B_{OPH}	B_{IH}	B_{IXCE}	B_{FEL}	B_{FEL}	B_{CCL}	B_{CCS}	B_{CCP}	
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	
2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	A000
3	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	$A_I A_5 A_4 A_3 A_2 A_1 A_0$ $\overline{P}$ A_F A $P_3 P_2 P_1 P_0$
4																	0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
5																	0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0
6																	0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1
7																	0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1
8																	0 1 0 1 0 1 0 0 0 1 1 0 0 1 0 1
9	0	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1	
																	C000 V
10				Sense													0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
																	A000
11																	0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 1
																	C000
12				Sense													0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	A000
14																	0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1
15				After verifying all inputs													0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
16				After verifying all P-terms													
17	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	
18	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
20																	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4.12. Code to Verify that Input Five in P-terms 20 is Virgin.

address for a product. As before each product is considered in the outer loop (index J). The inner loop (index K) pertains to the outputs F_0, --- F_7. If product P_J is used, and not contained within the output F_K, the fuse connecting the AND (product) gate and the (sum) gate is burned. The SUM flowchart appears in Fig. 4.13. The procedure starts by initializing PLA pin (Fig. 4.14). Afterwards addressing of the P-term is performed. The program will look for those used output functions which are not connected to the P-term and burn their links with the AND matrix. Then it initializes to perform verification that the respective output has been disconnected from the P-term. If the verification results are not satisfactory, this information is to be printed later. After checking all P-terms, the PLA pins are reset.

The coding in Fig. 4.15 is used to burn the fuse connecting output F_4 to P-term 26. The steps 1, 2, and 3 are sent to initialize the system for SUM, in accordance with Fig. 4.14. Now P-term (26) is addressed by means of steps 4 and 5. Afterwards, output F_4 is selected via step 6. Burning starts with step 7, which will apply V_{OPF} to output F_4, V_{FEH} (17.5V) to FE, then $\overline{CE}$ is pulsed to V_{IX} (10.5V) for about 0.5 msec. Steps 10, 11 and 12 are used to reset the Sum burn-in. Verification starts with step 13. Sensing is accomplished by step 14. Afterwards, verification is reset and the system will return to burn-in, by steps 16 and 17. When all outputs requiring burn-in have been considered and verified for P-term 26, the address of P-term 26 is removed. When the P-terms are finished, reset signals shown in steps 20, 21, 22, and 23 are sent to the hardware interface.

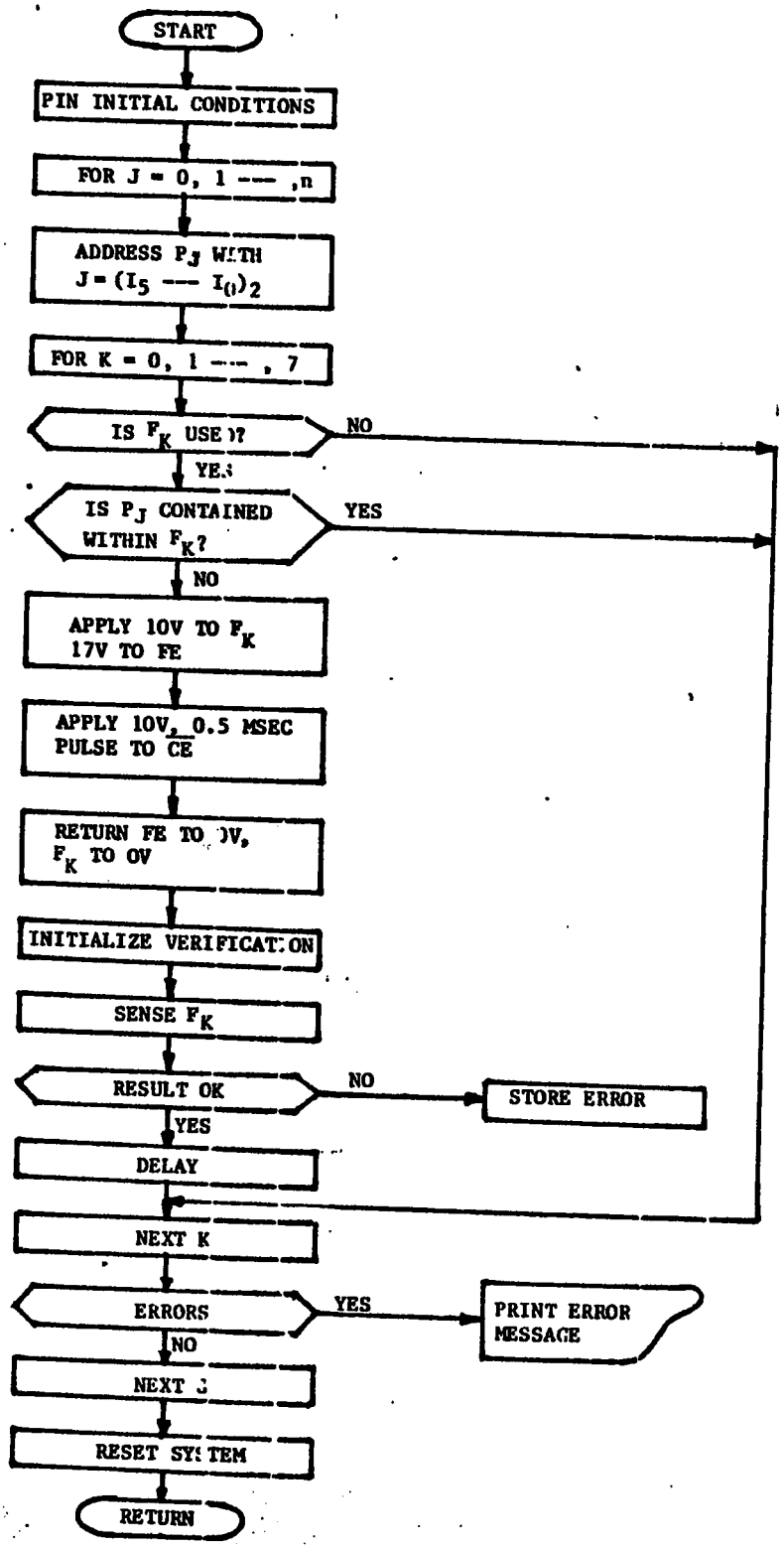


Figure 4.13 Flow Chart for Sum Burn-in and Its Verification (SUM).

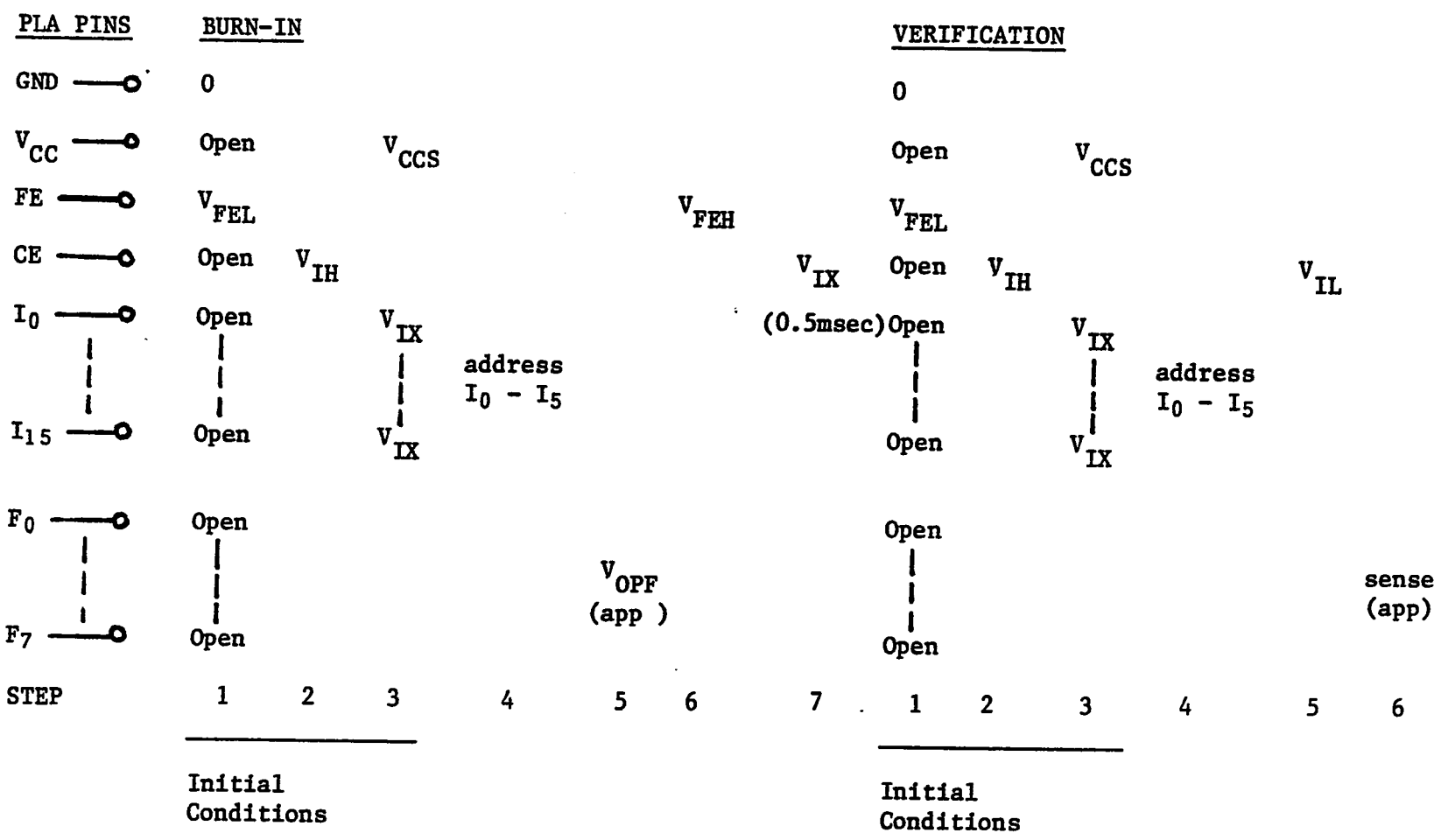


Figure 4.14 PLA Pin Input Sequence for Sum Burn-in and its Verification .

STEP	DATA																DATA
	9000																
		B_{ICE} / B_{sen7}		B_{IX} / B_{CE}		B_{sense}	B_{OPF}	B_{OPL}	B_{OPH}	B_{IH}	B_{IXCE}	B_{FEL}	B_{FEH}	B_{CCL}	B_{CCS}	B_{CCP}	
1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
2	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	A000
3	0	1	0	1	1	0	0	0	0	0	0	1	0	0	1	0	$A_I A_5 A_4 A_3 A_2 A_1 A_0$ $\overline{P}$ A_F A $P_3 P_2 P_1 P_0$
4																	0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0
5																	0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0
6																	0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0
7	0	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0	
8	0	1	0	1	1	0	1	0	0	0	0	1	1	0	1	0	
9	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	
DELAY						Programmed delay of 0.5 msec.											
10	0	1	0	1	1	0	1	0	0	0	0	1	1	0	1	0	
11	0	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0	
12	0	1	0	1	1	0	0	0	0	0	0	1	0	0	1	0	
13	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	
14	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	
																	C000 V
15						Sense											0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	
17	0	1	0	1	1	0	0	0	0	0	0	1	0	0	1	0	
DELAY						Programmed delay of 750 sec											
18						After burning output											A000
19						After burning P-terms											0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0
20	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0
21	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
23																	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4.15 Burning the Fuse Connecting Output F_4, to P-term 26 by "SUM".

The time function for the SUM program is:

$$F(J) = 250 \times 10^{-6} + 1.5 \times 10^{-3}(J) \quad \text{sec.}$$

Where J is the number of fuses to be burned.

4.4.4.2 Verification "SUMV"

The flowchart of this subrogram is shown in Fig. 4.16. The verification of sum starts by initializing PLA pins Fig. 4.14. Then an iterative loop will take place. The function of the loop is to verify each output for the P-terms on the chip. The verification subprogram is to verify that the sum gate connections to AND gates are still intact. If the program finds that some connections are burned it will print error messages. After verifying all P-terms the PLA pins will receive a reset signal.

The coding shown in Fig. 4.17 verifies the link of output F_1 with P-terms 6. It starts by initializing the system in accordance with Fig. 4.14 by sending codes of steps 1, 2 and 3. Then it addresses P-term 6 during steps 4 and 5. Afterwards, the address of output F_1 is provided by step 6. Verification starts by lowering $\overline{CE}$ to V_{IL} in step 7. Step 8 initiates sensing. After output F_1 is sensed, the system is reset to its initial condition. When all output functions are verified for P-term 6, the address of P-term 6 is removed by step 13. Upon completion of all P-terms verifications, the system is reset by steps 14, 15, 16 and 17.

The time required to verify all fuses in the sum matrix is about 38.5 msec.

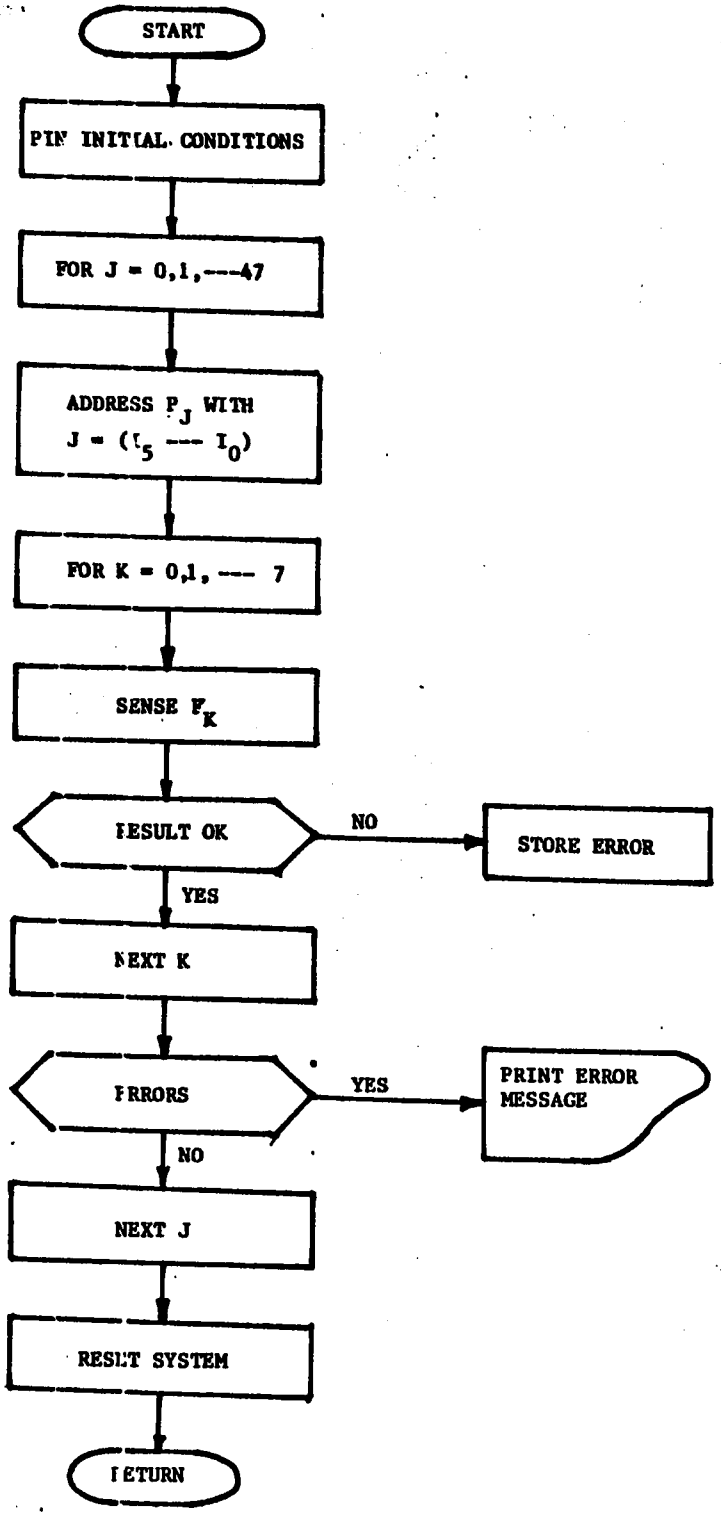


Figure 4.16 Flow Chart of Sum Verification "SUM V".

STEP	DATA																DATA
	9000																
		B_{ICE}	$B_{\overline{sen7}}$	$B_{\overline{IX}}$	$B_{\overline{CE}}$	B_{sense}	B_{OPF}	B_{OPL}	B_{OPH}	B_{IH}	B_{IXCE}	B_{FEL}	B_{FEH}	B_{CCL}	B_{CCS}	B_{CCP}	
1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
2	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	A000
3	0	1	0	1	1	0	0	0	0	0	0	1	0	0	1	0	$A_I A_5 A_4 A_3 A_2 A_1 A_0$ $\overline{P}$ A_F A $P_3 P_2 P_1 P_0$
4																	0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
5																	0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0
6																	0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1
7	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	
8	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	
																	C000 V
9						Sense											0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	
11	0	1	0	1	1	0	0	0	0	0	0	1	0	0	1	0	A000
12						After verifying output											0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0
13						After verifying P-terms											0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
14	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	
15	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
17																	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4.17 Verification of the Link of Output F_1 with P-term 6 by "SUMV"

4.4.5 Error Messages

When the program encounters an error, subroutine ERST will be utilized to store the error that has been discovered. Also, ERST will increase the error indicator location (010A) by one. Afterwards, subroutine PRINT will print the following message, unless the quantity of errors is zero:

ERROR ***

The specific error is printed by one of three sub-programs:

1. EOUT - This subprogram will print a message such as the following example:

OUTPUT 1,2,

This message indicates that an error in verifying outputs F_1 and F_2 has been encountered.

When this message appears from the OUT subprogram, it means fuses for outputs F_1 and F_2 have not been burned. The same message means that outputs F_1 and F_2 have burned fuses, if it appears from subprogram OUTV .

2. EPRD - The following is the message printed by EPRD: (PRODUCT) (P-term number X) (INPUTS) {(I_K) (Condition of I_K); for $0 \leq K \leq 15$}

This message means that an error is encountered in verified P-term of the product procedure. The input conditions are:

Uncomplemented	I
Complemented	C

Don't care D

Both fuses are present B

For example, consider the following message:

PRD P-02 I 0I, 1D, 2C, 3B,

The preceding indicates that in P-term 2, input variable I_0 is uncomplemented, I_1 is a don't care, and I_3 has both fuses intact.

3. ESUM - The message below is printed by ESUM to indicate that during verification, the P-term contained an error in the links of outputs F_J's:

(SUM) (P-term number X) (Outputs) (F_J) $0 \leq J \leq 7$

As an example consider the following message:

SUM P-05 OUTPUT 5,6,7,

This means that there is an error in verifying the links of outputs 5, 6, and 7 of the P-term number 5. If this message appeared as a consequence of SUM , it means that the stated outputs links are not burned. If it had appeared from SUMV , it means that the output links shown are burned.

Figure 4.18 illustrates other error messages examples.

4.4.6 Minor Programs

There are several other subprograms in the system. These subprograms are briefly summarized in Table XIV.

```
>VO ERROR ***  OUTPUT Ø,1,2,4,5,6,7,

>
 VP ERROR *** PRD P-ØØ IØI,1C,2C,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø1 IØC,1I,2C,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø2 IØC,1C,2I,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø3 IØC,1C,2C,3I,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø4 IØC,1C,2C,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-13 IØI,1C,2D,3C,4D,5D,6D,7C,8D,9D,AD,BD,CD,DD,EI,FD,
 ERROR *** PRD P-23 IØD,1D,2D,3C,4D,5D,6D,7D,8I,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-24 IØD,1D,2D,3C,4D,5D,6D,7D,8I,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-25 IØD,1D,2D,3C,4D,5D,6D,7D,8I,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-26 IØD,1D,2D,3D,4C,5D,6D,7D,8D,9D,AD,BD,CI,DD,ED,FD,
 ERROR *** PRD P-29 IØC,1C,2C,3C,4C,5C,6I,7C,8C,9I,AC,BC,CD,DD,ED,FD,
 ERROR *** PRD P-2C IØI,1C,2I,3C,4D,5D,6D,7D,8D,9D,AC,BC,CC,DC,EC,FC,

>VS ERROR *** SUM P-ØØ OUTPUT 2,3,4,5,6,7
 ERROR *** SUM P-Ø1 OUTPUT 2,3,4,5,6,7,
 ERROR *** SUM P-Ø2 OUTPUT 2,3,4,5,6,7,
 ERROR *** SUM P-Ø3 OUTPUT 2,3,4,5,6,7,
 ERROR *** SUM P-Ø4 OUTPUT Ø,3,4,5,6,7,
 ERROR *** SUM P-12 OUTPUT 2,
 ERROR *** SUM P-15 OUTPUT 1,3,
 ERROR *** SUM P-2Ø OUTPUT Ø,1,2,
 ERROR *** SUM P-26 OUTPUT Ø,1,2,
 ERROR *** SUM P-2C OUTPUT 1,2,
```

Figure 4.18 Error Messages

TABLE XIV Minor Programs in the Algorithm.

ROUTINE NAME	OPERATION
OVER	Used to establish verification conditions for OUT and OUTV. Also senses the specified output.
BURN	Used to burn the input of a given P-term by applying burn-in pulses for PROD.
PVER	Used to establish verification conditions, sensing, for PROD and PRODV.
RESET	Used to remove the address of a given input. Verifies results from sensing and stores errors, if there is any for PROD and PRODV.
SM1	Used to set initial conditions for SUM and SUMV.
SM2	Used to address a given P-term for SUM and SUMV.
SVER	Used to establish verification conditions, sensing for SUM and SUMV.
INC3	Will increment the value in location 104 by one. If this value is greater than 47, it causes a jump.
INC2	Will increment the contents of location 100 by one. If the value is greater than 15, it will cause a jump.
INC1	Will increment the contents of location 100 by one and if this value is greater than 7, it will cause a jump.

TABLE XIV Minor Programs in the Algorithm (Continued).

ROUTINE NAME	OPERATION
SENCH	Will examine if the contents of location 1Q5 is one and cause a jump, othereise it will return "no jump".
CHECK	Will examine if accumulator AC0 bit zero is one and will cause a jump, if so.
PRINT	If location 10A does not contain zero, it will print ERROR ***.
ESUM	Will print the following message: SUM P-nn OUTPUT a,b,c,... Where: nn is the P-term number stored in location 104, a,b,c are output numbers stored in location 10B-112.
EPRD	Will print the following message: PRD P-nn IaJ, bJ, ... Where: nn is the P-term number stored in location 104, a, b, c are input numbers stored in bits 0-3 of locations 10B-11A, J is a code to represent input status as follows: I uncomplemented, C complemented D don't care B both fuses intact. Storage of J is in bits 4-7 of locations 10B-11A.

TABLE XIV Minor Programs in the Algorithm (Continued).

ROUTINE NAME	OPERATION
EOUT	Will print the following message: OUTPUT a, b, c Where: a,b,c are output numbers stored in locations 10B-112.
PRPT	Will print the P-term numbers.
ERST	Will store the quantity of errors in location 10A. The PLA elements causing the errors are stored in consecutive locations, starting with 10B.
SENDA	Will print data in ASCII characters stored in AC0. Accumulator AC3 points to the original memory location of the ASCll character.
SHIFT 3	Will shift the contents of locations 101,102,103 one bit to the right.
SHIFT 2	Will shift the contents of location 10E and 101 one bit to the right.
SHIFT 1	Will shift the contents of location 101 one bit to the right.
SENDN	Will send the number stored in AC0 to the TTY for printing.

TABLE XIV Minor Programs in the Algorithm (Continued).

ROUTINE NAME	OPERATION
SENSE 1	Will sense a specified PLA output and store the result in location 105.
SENSE 2	Will sense a specified PLA output and store the result in location 106.
FIN	Will send reset signals to system through addresses 9CC0 and A000.

V. HARDWARE OVERVIEW AND MICROCOMPUTER INTERFACE

5.1 HARDWARE OVERVIEW

The block diagram of the hardware section is shown in Fig. 5.1. As described earlier, the hardware is divided into two major sectors; the microcomputer interface and the PLA interface. The PLA interface is implemented in CMOS logic gates, while the microcomputer interface is implemented in TTL gates. Three boards were used to build the hardware, using available parts. The hardware accepts 16-bit input and can be linked to any system which can provide such inputs. The hardware recognizes addresses 9000 and A000 for its input and uses address C000 as an output to the microcomputer. It is also wired for a PLA of 28 pins. The PLA pins consist of 16 inputs (having 0, 5V, 10V or open voltage levels), 8 outputs, (having 0, 5V, 10V, 17V, or open voltage levels), and 4 pins for the remaining connections.

5.2 MICROCOMPUTER INTERFACE

All microcomputer signals, interconnecting the interface, are with respect to the LCDS interface bus. Should the signals be microcomputer outputs, addresses arrive first and are latched, then decoded. The decoded output addresses of 9000 or A000 are used to latch output data and input address C000 controls the tri-state buffer input to the microcomputer. The TTL levels are converted to CMOS levels, because CMOS levels are needed in the

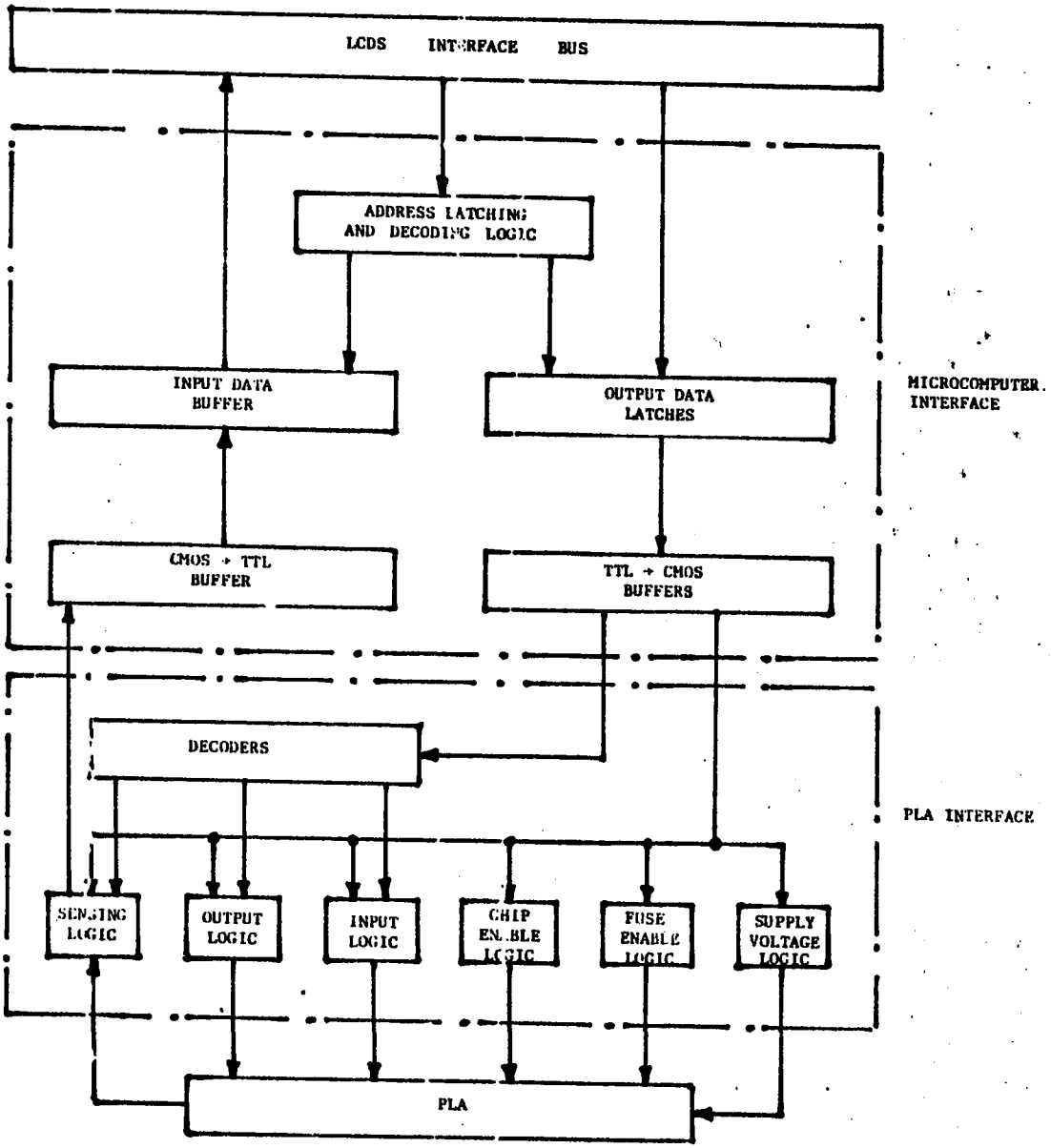


Figure 5.1 Hardware Block Diagram.

PLA interface circuit. The output of the sensing function, which is used to verify required outputs, is transferred to TTL levels as required for microcomputer inputs. The sensing function is applied to the microcomputer only if address C000 is first received from the LCDS interface bus.

The upper part of the hardware block diagram (Fig. 5.1) shows the sections of the microcomputer interface.

5.2.1 Microcomputer Address/Data Bus

The bidirectional tranceiver elements are used to buffer the 16-bit, time multiplex address/data bus for up to 30 TTL loads. This buffered parallel input/output bus has the address and data multiplexed on it in the form of an address output followed by a data transfer interval |13|.

A special set of strobes determines whether the microcomputer is to provide an output or anticipates an input. The LCDS address/data transfers are synchronized by a: (1) negative buffered address strobe (NBADS) for addresses, (2) buffered output data strobe (BODS) for output data, and (3) buffered input data strobe (BIDS) for input data. The signals NBADS, BODS, and BIDS are generated by the microcomputer and are shown in Fig. 5.2 |15|. Signal NBADS is used to latch address data because it appears in the middle of the valid address data time interval. Output data is latched using the trailing edge of BODS which appears in the portion where output data is valid. Input data is valid around the trailing edge of BIDS. Hence the clock pulses surrounding the trailing edge of

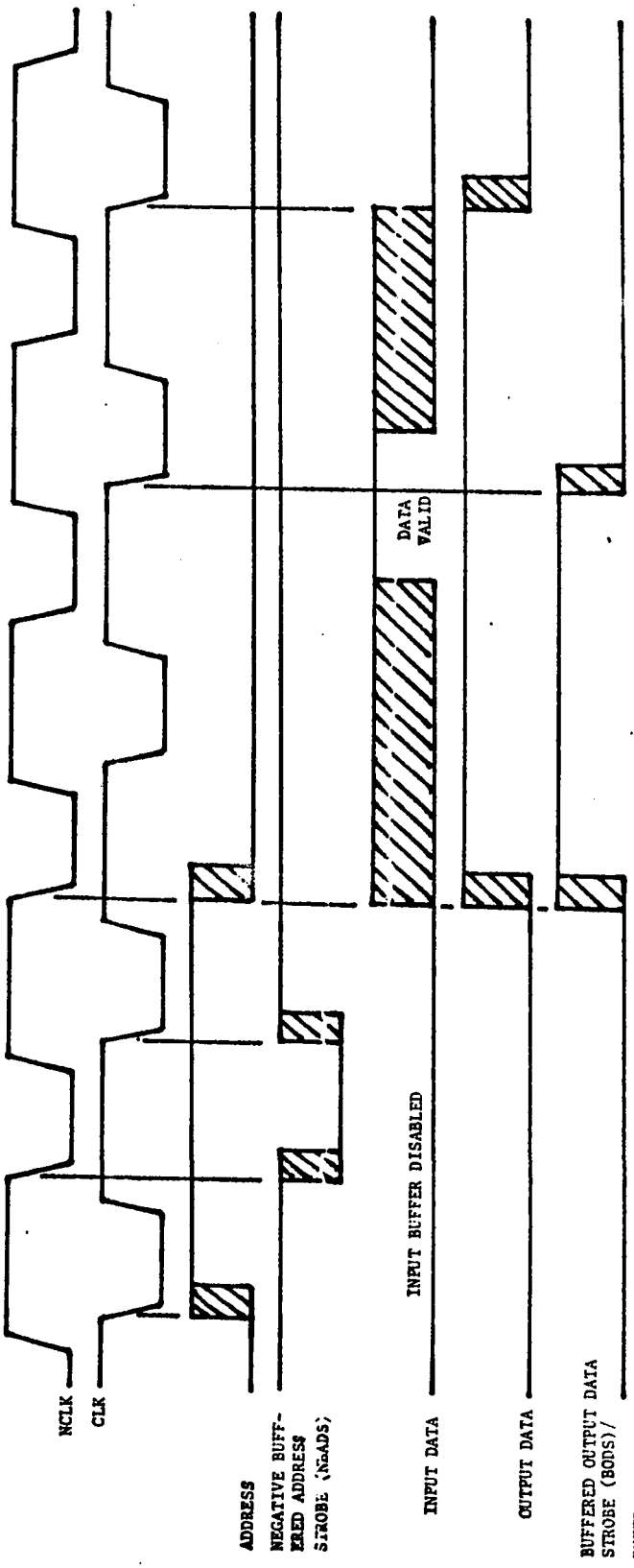


Figure 5.2 I/O Signal Strobes.

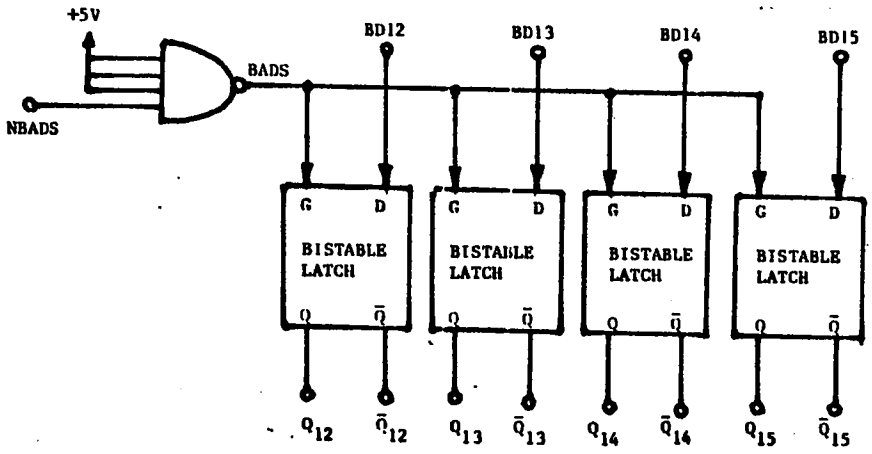
IDS are chosen to signal the control of a tri-state buffer for transferring data from the PLA interface to the microcomputer.

5.2.2 Address Latching and Decoding

The circuit in Fig 5.3a is used to distinguish the addresses from the multiplexed address/data output. As mentioned before two addresses are used for microcomputer output data; 9000 and A000, and address C000 is used for microcomputer input data. Only bits 12 through 15 are needed for decoding these addresses. This is because the off-board address space is between addresses 1000 and EFFF. Since the microcomputer service programs do not use the off-board addresses, the only signals arriving to these addresses pertain to peripherals. Therefore, when bits 15 and 12 are high and bit 14 is low the address ranges are 9000 to 9FFF or B000 to BFFF. This means if a peripheral uses any of the above mentioned addresses, it needs sense only the state of these three bits. Address A000 was actually chosen from the ranges A000 to AFFF and B000 to BFFF. Address C000 was chosen from the ranges C000 to DFFF.

The circuit shown in Fig. 5.3a is used to latch the address data. As it is evident from the Fig. 5.3c waveforms, when NBADS becomes low in the middle of the address data, the output of the latch (Q) follows the input data. When NBADS is high, Q maintains its present state. The functional table of the latch used is shown in Fig. 5.3b.

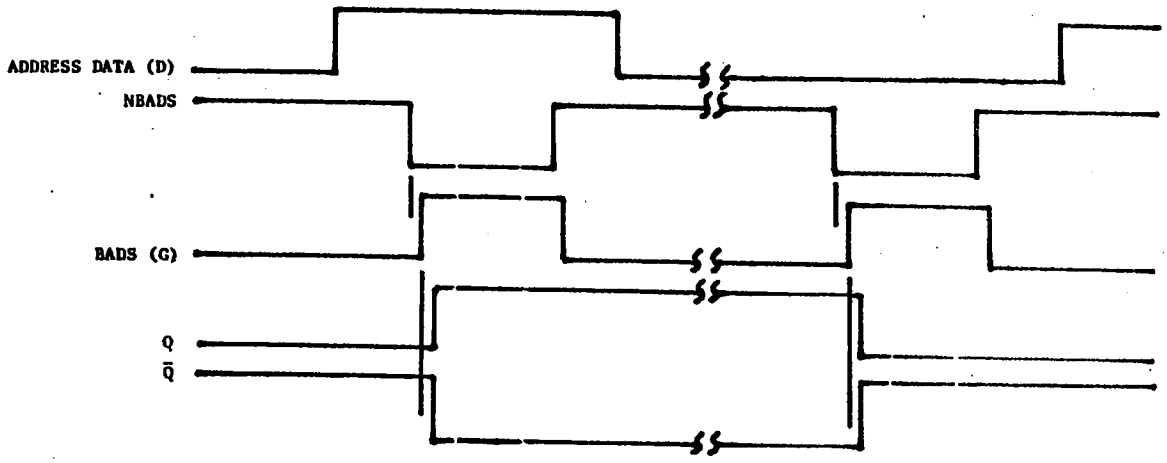
The circuit shown in Fig 5.4a is used to decode the addresses.



a. Circuit Used to Latch the Address Data

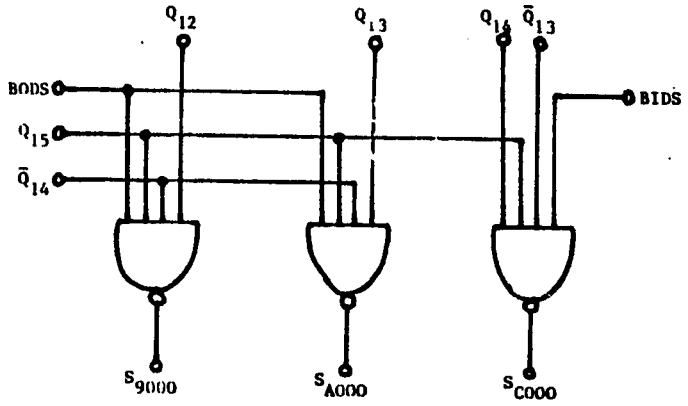
D	G	Q	$\bar{Q}$
L	H	L	H
H	H	H	L
X	L	Q_o	$\bar{Q}_o$

b. Functional Table of the Bistable Latch.

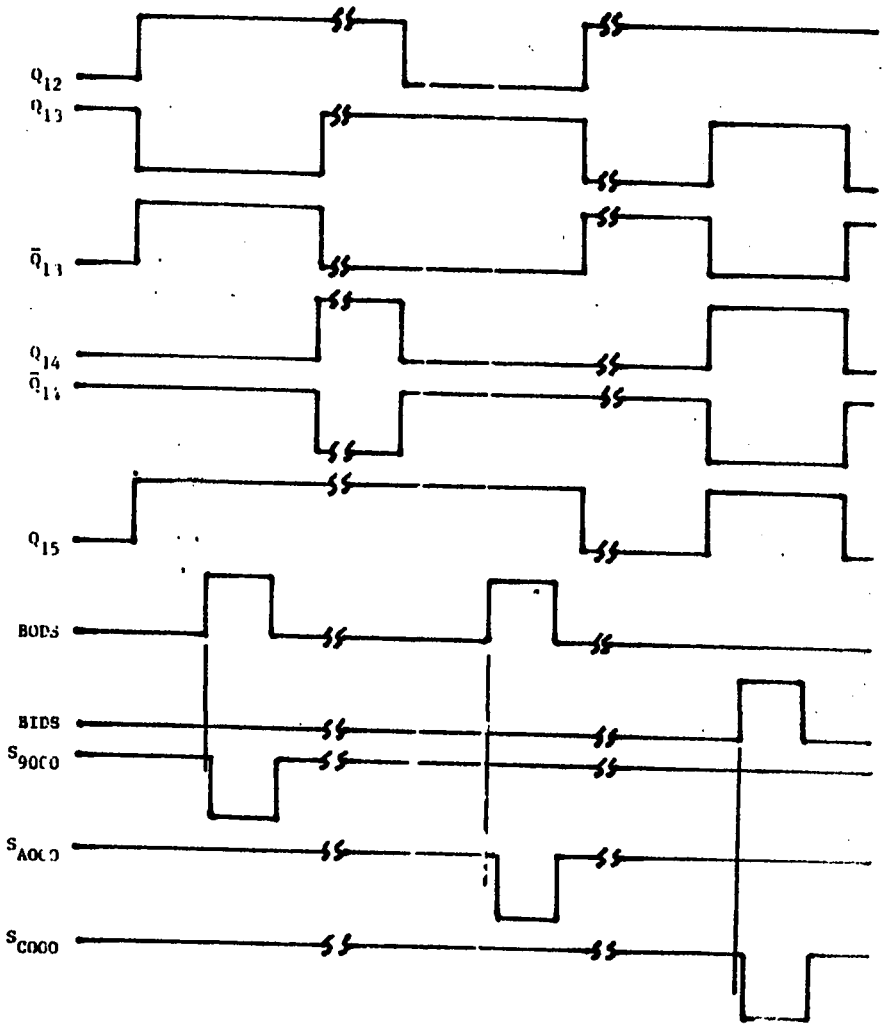


c. Waveform Diagram of a Bistable Latch

Figure 5.3 Circuit and Waveforms of Address Data Latching.



a. Circuit Used to Decode Latched Address Data.



b. Waveforms of the Circuit in Part (a).

Figure 5.4 Circuit and Waveforms of Address Decoding.

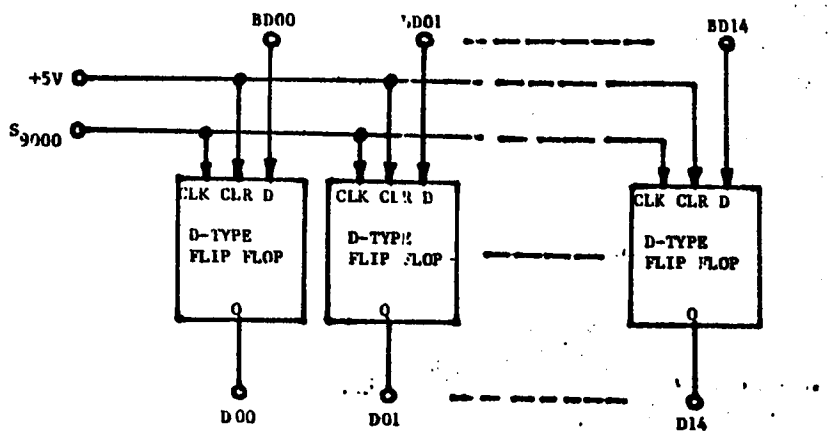
In the circuit the bits needed for each address are fed to a decoder (NAND gate array). The waveforms of Fig. 5.4c indicate that, for address 9000, Q15, $\overline{Q14}$ and Q12 are simultaneously high. Should BODS be high, the signal for latching the data of address 9000 (S_{9000}) assumes a low logic level. The signal S_{9000} will follow BODS and it will go high when BODS goes low. Similarly, the signal for latching the data of address A000 (S_{A000}) will go low when BODS, Q15, $\overline{Q14}$ and Q13 are high. For the input address, its latching signal will be the complement of IDS, provided that address data signals Q15, Q14 and $\overline{Q13}$ are high.

5.2.3 Output Data Latching

Because the trailing edge of BODS is in the region of valid output data, this time instant can be used as the clock trigger for the data latches. The circuit shown in Fig 5.5a is used to latch data bits 00 through 14 for address 9000. A similar circuit is used to latch output data bits 00 through 14 for address A000, but with S_{A000} as the CLK signal. As shown in the waveforms Fig 5.5c, the output data is latched when the CLK has a low to high transition. Output data bits 5 and 13 are shown as an example of latching the output data at address 9000. Figure 5.4b shows the functional table of the D-type flip-flop used in the circuit to latch output data.

5.2.4 Input Data

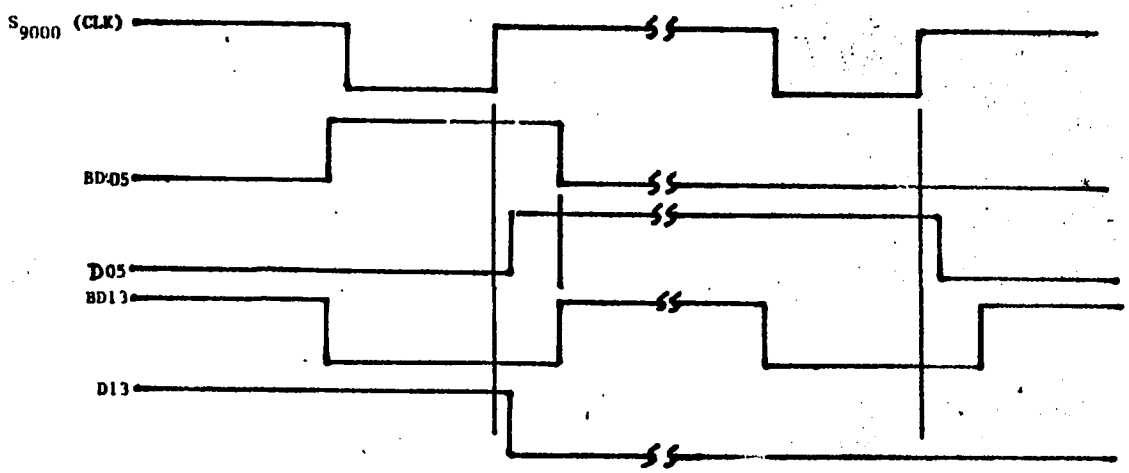
Tri-state buffers are used in the input data path to the



a. Circuit Used to Latch Data for Address 9000.

CLR	CLK	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_o
H	H	X	Q_o

b. Functional Table of the D-TYPE FLIP FLOP Used in Part (a).



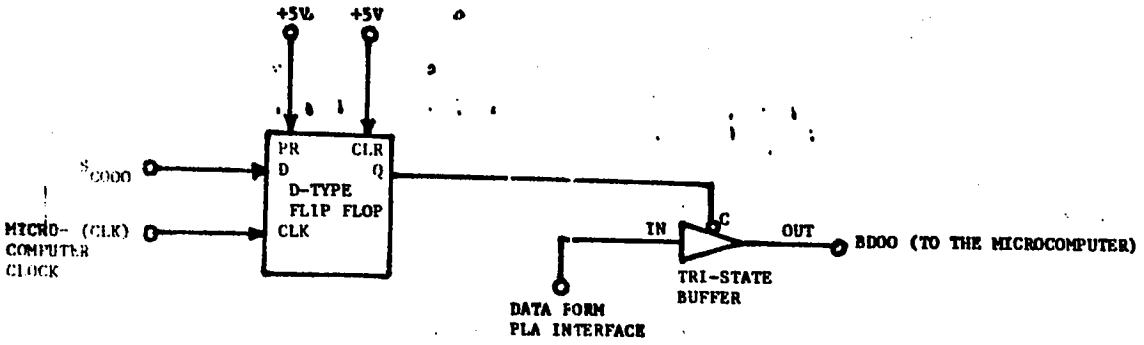
c. Waveforms Showing Data Latching of BD05 and BD13 in Address 9000.

Figure 5.5 Circuit and Waveforms Used to Latch Data for Address 9000.

microcomputer. The tri-state buffers are required because many sources utilize the LCDS interface bus. Therefore, only one source at a time can have a bus connection. The circuit shown in Fig 5.6a is used to signal that the input to the microcomputer is allowed. The waveforms for the circuit are shown in Fig 5.6d. Since the data input is valid around the trailing edge of BIDS, the upward transision of the CLK is used to signal (clock) the control of the tri-state buffer. The S_{C000} signal will go low when BIDS goes high. Then Q will follow D (S_{C000}), when an upward transition of the CLK exists. Hence, Q will go low when D is low and it will go high when D is high. The functional table of the D-type flip-flop used here is shown in Fig 5.6b.

5.2.5 TTL-CMOS Interface

The CMOS technology is used in the PLA interface circuits. The reason CMOS was chosen is that it can operate with the high voltage levels (e.g. 17V) required by the PLA burn-in procedure. Standard TTL gates are limited to 5 V. However, when interfacing CMOS and TTL, voltage levels and current values need to be taken into consideration. Table XV compares CMOS and TTL input and output specifications for a power supply voltage between 4.5 V and 5.5 V [17]. It is clear from the table that the guaranteed TTL output voltage of 2.4 V is lower than the minimum CMOS input voltage required to guarantee switching, i.e. 3.5 volts. The use of a pull-up resistor, R_A in Fig 5.7a, compensates for the differ-



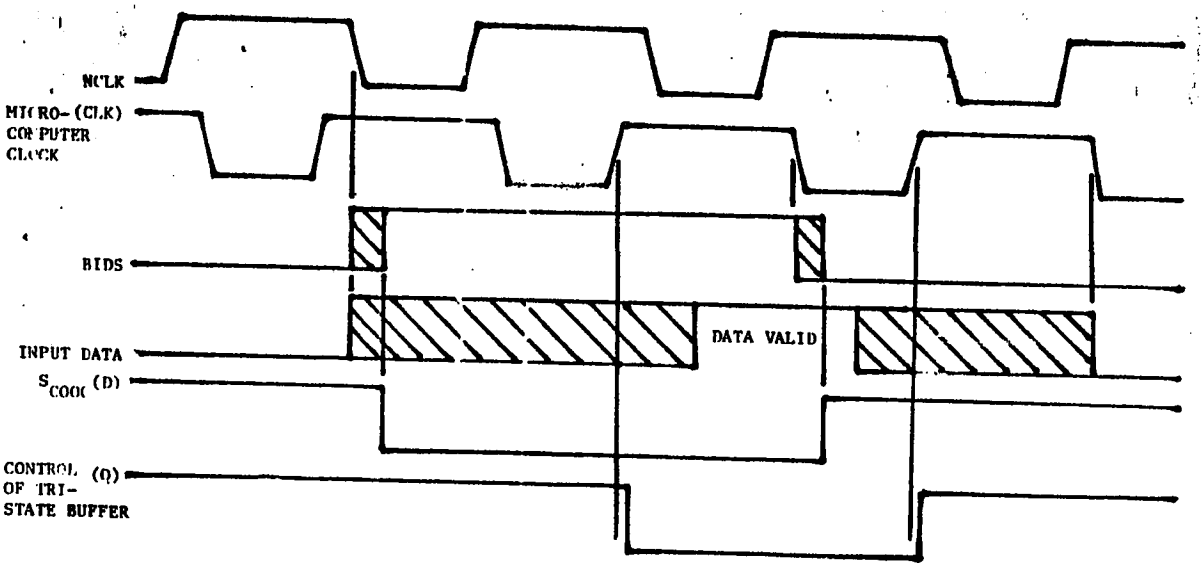
a. Circuit Used to Transfer Data to Microcomputer.

PR	CLR	CLK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	TOGGLE	
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_o	$\bar{Q}_o$
H	H	H	X	Q_o	$\bar{Q}_o$

b. Functional Table of the D-TYPE FLIP FLOP Used in Part (a).

IN	C	OUT
L	L	L
H	L	H
X	H	OPEN

c. Functional Table of the Tri-state Buffer Used in Part (a).

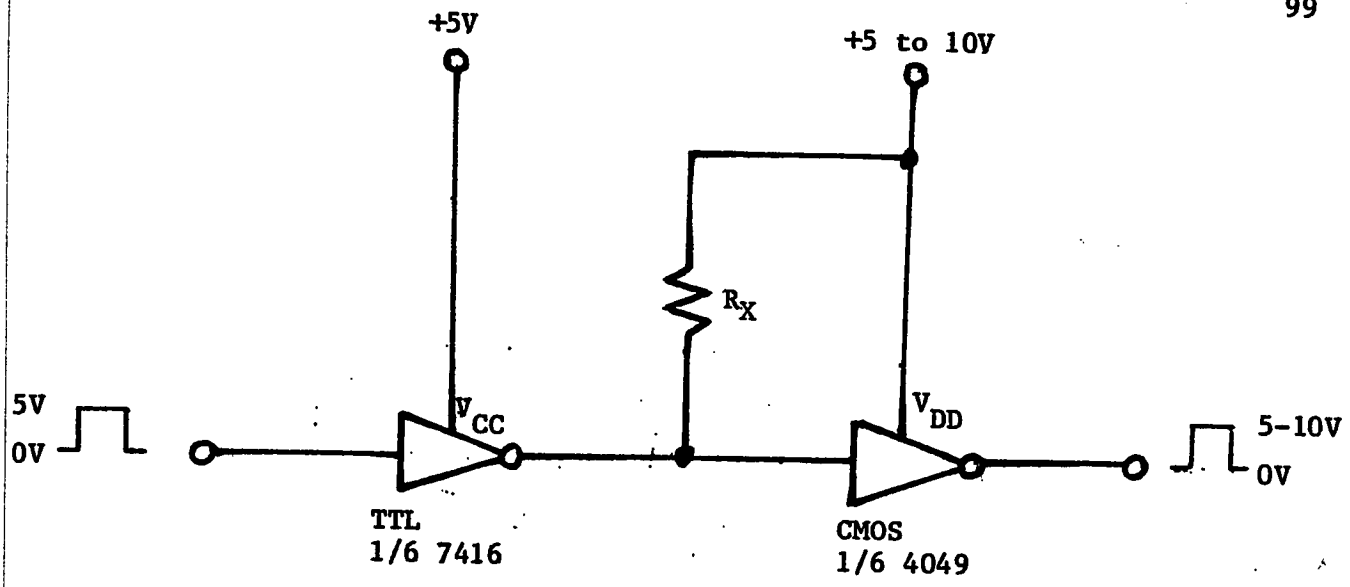


d. Waveforms of Data Input.

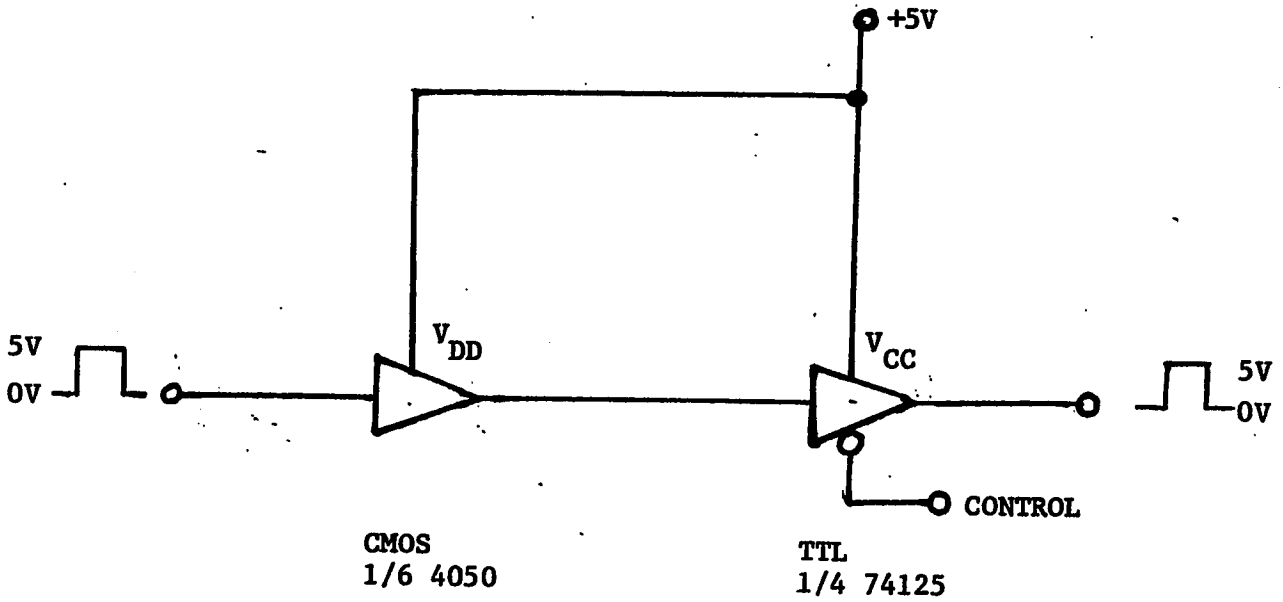
Figure 5.6 Data Input to the Microcomputer.

TABLE XV TTL and CMOS Input and Output Specifications at a Supply Voltage Between +4.5V and 5.5V.

PARAMETER	SYMBOL	TTL	CMOS	UNIT
Maximum voltage level for logic-0 input	V_{IL}	0.8	1.5	V
Maximum voltage level for logic-0 output	V_{OL}	0.4	0.5	V
Maximum voltage level for logic-1 input	V_{IH}	2	3.5	V
Maximum voltage level for logic-1 output	V_{OH}	2.4	4.5	V
Maximum input source current at logic 0	I_{IL}	-1.6×10^{3}	-1	µA
Maximum output sink current at logic 0	I_{OL}	16	1.7	m A
Maximum input sink current at logic 1	I_{IH}	40	1	µA
Maximum output source current at logic 1	I_{OH}	-400	-1.7×10^{3}	µA



a. TTL to CMOS Interface



b. CMOS to TTL Interface

Figure 5.7. CMOS - TTL Interface.

ence. The minimum value of R_X is fixed by the maximum sink current. The maximum resistance is determined by the "off" leakage current of the TTL output sink transistor. The value of R_X between 1.5 and 4.7 KΩ are suitable for all TTL families |15|. A value of 2.2 KΩ is used for the system described herein. When the V_{DD} supply voltage is greater than +5V, high voltage open-collector output TTL circuits, such as the 7416 gate, is used in conjunction with CMOS 4049 or 4050 buffers as shown in Fig. 5.7a. The value of the pull-up resistor R_X, for V_{DD} at 10 V, is 39 KΩ |19|.

In the CMOS to TTL interface, the requirement is to sink sufficient current in the low output state at a maximum output voltage of 0.4 volts |18|. The 4049 and 4050 buffers are used, as shown in Fig. 5.7b for this purpose. They have the advantage of accepting a maximum input voltage of 5 to 15 V and can provide outputs at TTL levels.

The system was therefore designed such that all latched data needed to be transferred to 10 volts CMOS levels use the interface circuit of Fig 5.7a, with R_X = 39KΩ. Data transferred to 5 volts CMOS levels utilize the circuit in Fig 5.7a, with R_X = 2.2 KΩ. The CMOS high voltage levels are transffered to TTL level by using circuit 5.7b.

5.3 MICROCOMPUTER INTERFACE ASSEMBLING

One PC board (board 1) was used for the microcomputer interface circuit. The schematic diagram for board 1 is shown in

Fig. 5.8. The ICs board locations are shown in Fig. 5.9. The I/O signals for the board are shown in Fig. 5.10. The parts list is provided by Table XVI. The notation (1, I) is used to indicate pin I of board 1.

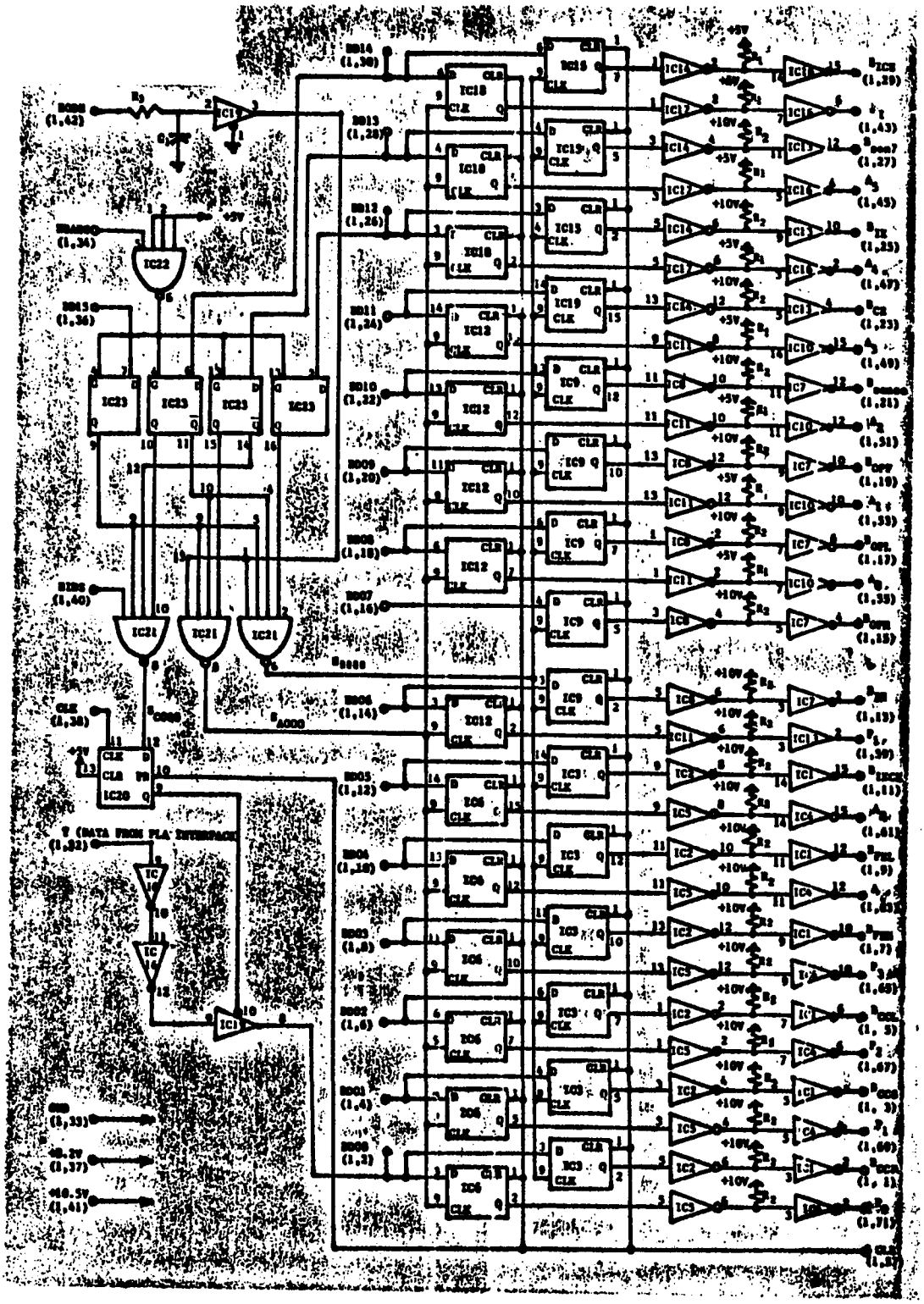


Figure 5.8 Microcomputer Interface Circuit (Board 1)

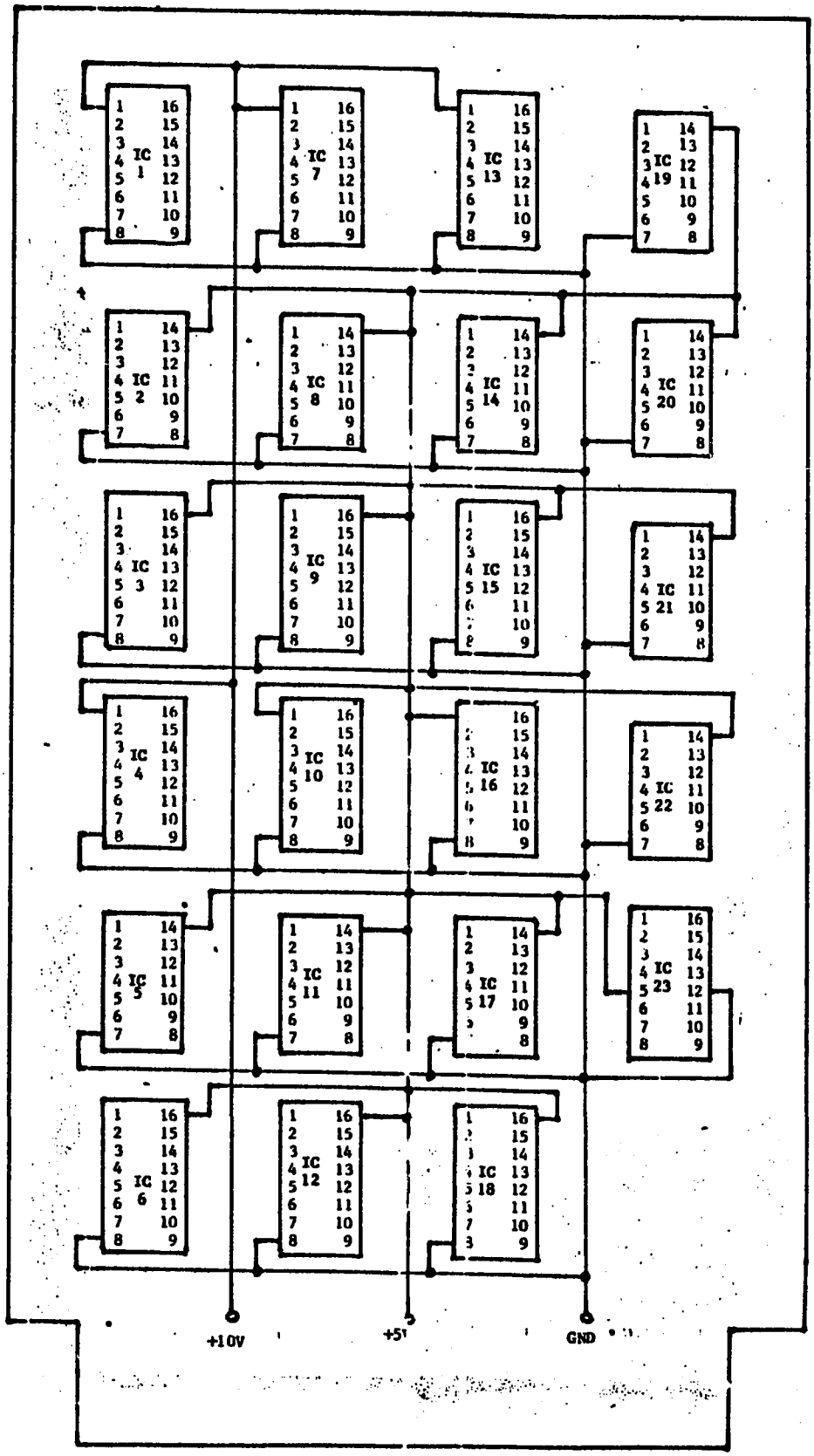


Figure 5.9 IC Organization for Board 1.

	FRONT	BACK		
B_{CCP}	1	2	BD00	
B_{CCS}	3	4	BD01	
B_{CCL}	5	6	BD02	
B_{PEH}	7	8	BD03	
B_{FEL}	9	10	BD04	
B_{IXCE}	11	12	BD05	
B_{IH}	13	14	BD06	
B_{OPH}	15	16	BD07	
B_{OPL}	17	18	BD08	
B_{OPF}	19	20	BD09	
B_{sense}	21	22	BD10	
$B_{\overline{CE}}$	23	24	BD11	
B_{IX}	25	26	BD12	
B_{sen7}	27	28	BD13	
B_{ICE}	29	30	BD14	
	31	32	T	Information from PLA Interface
GND	33	34	NBADS	
	35	36	BD15	
+5V	37	38	CLK	
	39	40	BIDS	
+10V	41	42	BODS	
A_I	43	44		
A_5	45	46		
A_4	47	48		
A_3	49	50		
A_2	51	52		
A_1	53	54		
A_0	55	56		
	57	58		
P	59	60		
A_F	61	62		
A	63	64		
P_3	65	66		
P_2	67	68		
P_1	69	70		
P_0	71	72		

(Note. Empty Locations : No connection)

Figure 5.10 I/O Signals for Board 1.

TABLE XVI Parts List for Microcomputer Interface (Board 1).

PART SYMBOL	PART #	DESCRIPTION	QTY
IC 6,12,18,3,9,15	SN74174	Hex latch	6
IC 2,8,14,5,11,17	SN7416	Hex inverter	6
IC 2,3	SN7475	Quad latch	1
IC 21,22	SN7420	Dual 4-input NAND	2
IC 20	SN7474	Dual latch	1
IC 19	SN74125	Quad tri-state buffer	1
IC 1,7,4,10,16	F 4049 PC	Hex inverter	5
IC 13	F 4050 PC	Hex buffer	1
R_2		39 KΩ resistor (1/4 watt)	21
R_1		2.2 KΩ resister (1/4 watt)	8
R_3		470 Ω resister (1/4 watt)	1
C_1		1000 pF capacitor (50V)	1
		72-pin socket	1
		72-I/O IC board	1

VI PLA INTERFACE

6.1 GENERAL

The PLA interface employs the CMOS logic family. As shown in Fig. 5.1, the PLA interface operates on signals generated by the microcomputer interface. These signals control the circuits which provide the voltage levels to the PLA pins. These voltage levels are defined by the manufacturer (Appendix A). The PLA pins are divided into 6 different sets indicated by Table XVII.

6.2 DESIGN AND ANALYSIS OF PLA INTERFACE CIRCUITS

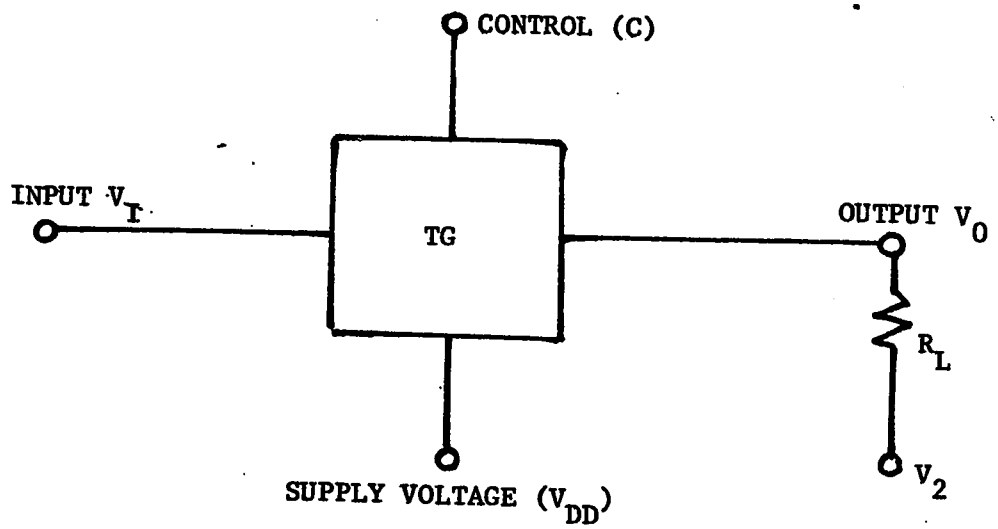
6.2.1 Transmission Gate

An important circuit element used by the PLA interface circuit, is the transmission gate (TG). Its symbol and representation are shown in Fig.6.1. When the control voltage is high TG will behave as a low valued resistance (r_{on}). When the control voltage is low TG will be essentially an open circuit (r_{off}). The off resistance (r_{off}) is very high ($1.8 \times 10^{11}\Omega$) and it needs very sophisticated equipment to measure it [19]. Because of its non-linear behavior the "off" leakage current instead is normally considered. A typical value of the leakage current when TG is off is about 100 pA at 18V. The TG is a bilateral device which means that its input can be used as an output and vice versa. Selected r_{on} values are shown in Table XVIII. An approximate relation for r_{on} is [18]:

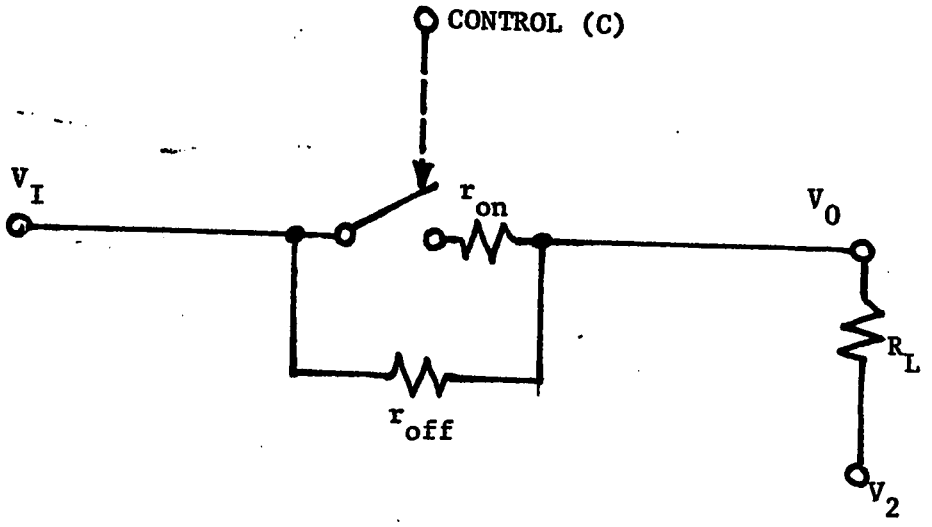
$$r_{on} = \frac{|V_I - V_0|}{|V_0 - V_2|/R_L}$$

TABLE XVII PLA Pins .

#	NAME	SYMBOL	PIN QTY
1	Chip Enable	$\overline{CE}$	1
2	Input Variable	I_K	16
3	Supply Voltage	V_{CC}	1
4	Fuse Enable	FE	1
5	Output Functions	F_J	8
6	Ground	G	1



a. Transmission Gate Symbol.



b. Transmission Gate Representation.

Figure 6.1. CMOS Transmission Gate.

TABLE XVIII Selected Values of r_{on}.

SUPPLY VOLTAGE V_{DD} (V)	INPUT VOLTAGE V_1 (V)	OUTPUT VOLTAGE V_0 (V)	V_2 (V)	LOAD R_L (Ω)	r_{on} (Ω)
10.50	10.50	10.16	0	1800	60.24
10.50	10.50	10.33	0	3600	59.24
10.50	4.51	4.51	0	82K	2.0 (measured)
10.50	0.0	0.02	5.01	10K	40.08
17.50	10.5	10.07	0	910	38.86
17.50	0.0	0.03	5.01	4.7K	28.31

6.2.2 Chip Enable ($\overline{CE}$) Circuits

The $\overline{CE}$ signal is to assume exactly one of the following states |12|:

1. V_{IX} : $9.5 \leq V_{IX} \leq 10.5$ V @ 5 mA
2. V_{IH} : $2.4 \leq V_{IH} \leq 5.5$ V @ 50 μA
3. V_{IL} : $0 \leq V_{IL} \leq 0.8$ V @ - 0.5 mA
4. Open: High impedance

To implement these states, the circuit in Fig. 6.2a was designed. When B_{IXCE} is high, $V_{\overline{CE}}$ will assume the voltage V_{IX}. Its value, due to loading, is calculated by the following equation (assuming other gates are off with negligible leakage current):

$$V_0 = V_{IN} - I_L r_{on} \qquad (6.1)$$

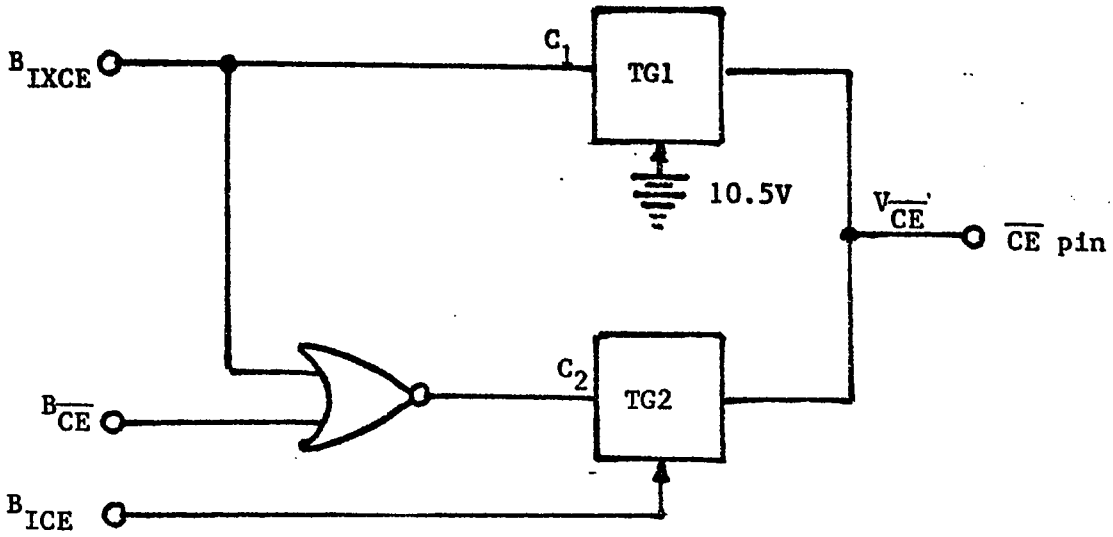
Where: V_0 in this case is $V_{\overline{CE}}$ and V_{IN} is 10.5V (B_{IXCE} high),
I_L is the load current specified by the manufacturer,
r_{on} is approximated from Table XVIII.

Substituting:

$$V_{\overline{CE}} \simeq 10.5 - 5 \times 10^{-3} \times 60 = 10.2V$$

The above is in the limits of the specified V_{IX}.

Also, when B_{IXCE} is high, the NOR gate prevents conflicting voltages from



a. $\overline{CE}$ Pin Circuit Diagram

B_{IXCE}	$B_{\overline{CE}}$	B_{ICE}	TG1	TG2	$V_{\overline{CE}}$
0	0	0	off	on	V_{IL}
0	0	1	off	on	V_{IH}
0	1	X	off	off	Open
1	0	X	on	off	V_{IX}
1	1	X	on	off	V_{IX}

b. Truth Table for $\overline{CE}$ Circuit in part (a)

Figure 6.2. Circuit Diagram and Truth Table of $V_{\overline{CE}}$

being applied to pin $\overline{CE}$. If $B_{\overline{CE}}$ and B_{IXCE} are both low TG2 is on, as evidenced from the truth table shown in Fig. 6.2b. For this case $V_{\overline{CE}}$ will assume the same value as the input B_{ICE}. If $B_{\overline{CE}}$ and B_{IXCE} are both low, then if B_{IXCE} goes high, $V_{\overline{CE}}$ will change from V_{IH} or V_{IL} to V_{IX}. The propagation delay, which temporarily allows a conflict, has been found to be negligible experimentally.

At B_{ICE} high, $V_{\overline{CE}}$ in accordance with equation 6.1, will equal:

$$V_{\overline{CE}} \simeq 5.2 - 50 \times 10^{-6} \times 2 = 5.2 \text{ V}$$

For B_{ICE} low:

$$V_{\overline{CE}} \simeq 0.05 - (-0.5 \times 10^{-3} \times 40) = 0.03 \text{ V}$$

Both values satisfy the V_{IH} and the V_{IL} limitations.

When B_{IXCE} is low and $B_{\overline{CE}}$ is high both gates are off. Hence, the $\overline{CE}$ pin input is connected to a high impedance path having a very small total leakage current. Two such leakage currents exist for the TG's and the total equals about 200 pA.

6.2.3 Input Variable (I_K) Circuits

The PLA input variable pins will be at one of the following states [12]:

1. V_{IH} : $2.4 \leq V_{IH} \leq 5.5$ V @ 50 μA

2. V_{IL} : $0 \leq V_{IL} \leq 0.8$ V @ -0.5 mA

3. V_{IX} : $9.5 \leq V_{IX} \leq 10.5$ V @ 2.5 mA

4. Open:-High impedance

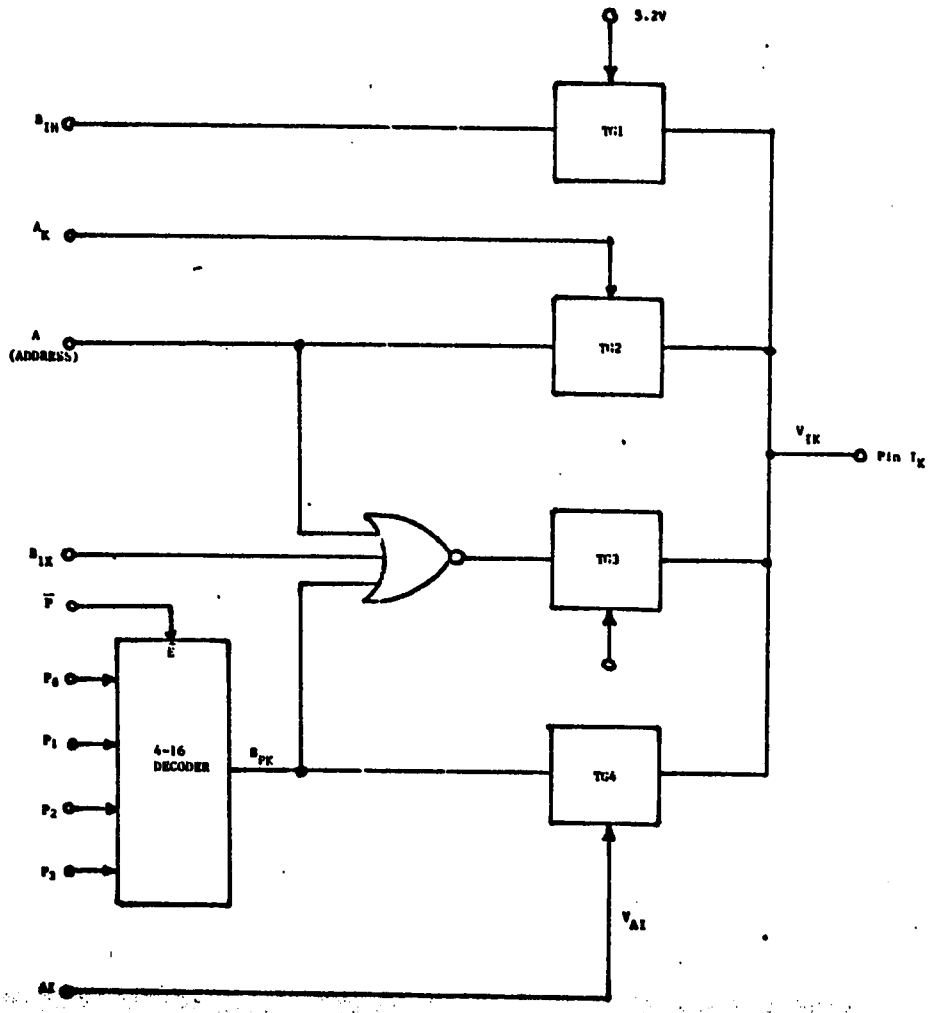
The circuit shown in Fig 6.3a is used to enact the states for each input variable. The address part of the circuit is applied only to inputs I_0 through I_5. Table XIX shows the relation between input signals and input variable and voltage (V_{IK}) levels. One state is selected at a particular time, using the software. A hardware protective circuit is also used here because it was simpler to accomplish than by software. For example, a conflict between B_{IX} and A is potentially possible. However, A is protected via the NOR gate shown. Conflicts between B_{IH} and other state signals are protected through software.

When B_{IH} is high, all input variables are at V_{IH} voltage level. The voltage at each input variable designated as is V_{IK} (K = 1, . . . , 16) and, in accordance with equation 6.1:

$$V_{IK} \simeq 5.2 - 50 \times 10^{-6} \times 2 = 5.2 \text{ V}$$

This value satisfies the V_{IH} limitations.

When A is high and B_{IX} is low, input variables I_0 through I_5 will assume the values of A_0 through A_5 (V_{IL} or V_{IH}) according to corresponding subscripts. Other variables are at V_{IX}. The



a. Circuit Used to Realize Input Variables States.

INPUT					OUTPUT			
P_0	P_1	P_2	P_3		B_{P0}	B_{P1}	----	B_{P15}
0	0	0	0		1	0		0
0	0	0	1		0	1		0
1	1	1	1		0	0		1
1	x	x	x	x	0	0		0

b. Partial Truth Table for 4 - 16 Decoder.

Figure 6.3 Input Variables Circuit Diagram.

TABLE XIX Input Variables Voltage Levels Selected Via Input Signals.

B_{IH}	A	B_{IX}	P	INPUT VARIABLE VOLTAGE LEVELS
0	0	0	0	V_{IX}
0	0	0	1	Selected I_K assumes the voltage level of A_I(V_{IH} or V_{IL}). Other inputs are at V_{IX}.
0	0	1	0	Open
0	0	1	1	Open (software protected, does not exist) (NA)
0	1	0	0	Inputs I_0-I_5 assume the values of A_0-A_5 (V_{IH} or V_{IL}). Others are at V_{IX}.
0	1	0	1	Open (NA)
0	1	1	0	Open (NA)
0	1	1	1	Open (NA)
1	0	0	0	Open (NA)
1	0	0	1	Open (NA)
1	0	1	0	V_{IH}
1	0	1	1	Open (NA)
	upto			
1	1	1	1	Open (NA)

circuit analysis for A high, provided B_{IX} is low, is similar to the generation of the $V_{\overline{CE}}$ signal for the values V_{IL} or V_{IH}.

If B_{IX} is low, provided that A is low (for variables I_0 through I_5) and B_{PK} is low for a given input variable, pin I_K will be at V_{IX}. Hence, the voltage applied to I_K using equation 6.1:

$$V_{IK} \simeq 10.5 - 2.5 \times 10^{-3} \times 60 = 10.35 \text{ V}$$

When B_{PK} is high, and B_{IX} is low, input variable I_K will assume the value of V_{AI} (V_{IL} or V_{IH}). Other input variables will be at V_{IX}. The 4-16 decoder is used to select which of the input variable will be at the voltage level of V_{AI}. The 4-16 decoder partial truth table is shown in Fig. 6.3b. The circuit analysis for B_{PK} high is similar to the generation of $V_{\overline{CE}}$ signal for the values V_{IL} or V_{IH}.

When B_{IH}, A, P are low and B_{IX} is high, the input variables will be at the open state, with leakage current of about 400 pA for variables $I_0 - I_5$ and 300 pA for variables $I_6 - I_{15}$.

6.2.4 Circuit Assemblies for Signals to Pins $\overline{CE}$ and I

The $\overline{CE}$ and I circuits are built on one board (board 2). The circuits schematics is shown in Fig. 6.4. The chip organization and I/O signals to board 2 are shown in Figures 6.5 and 6.6 respectively. All parts used in board 2 are listed in Table XX.

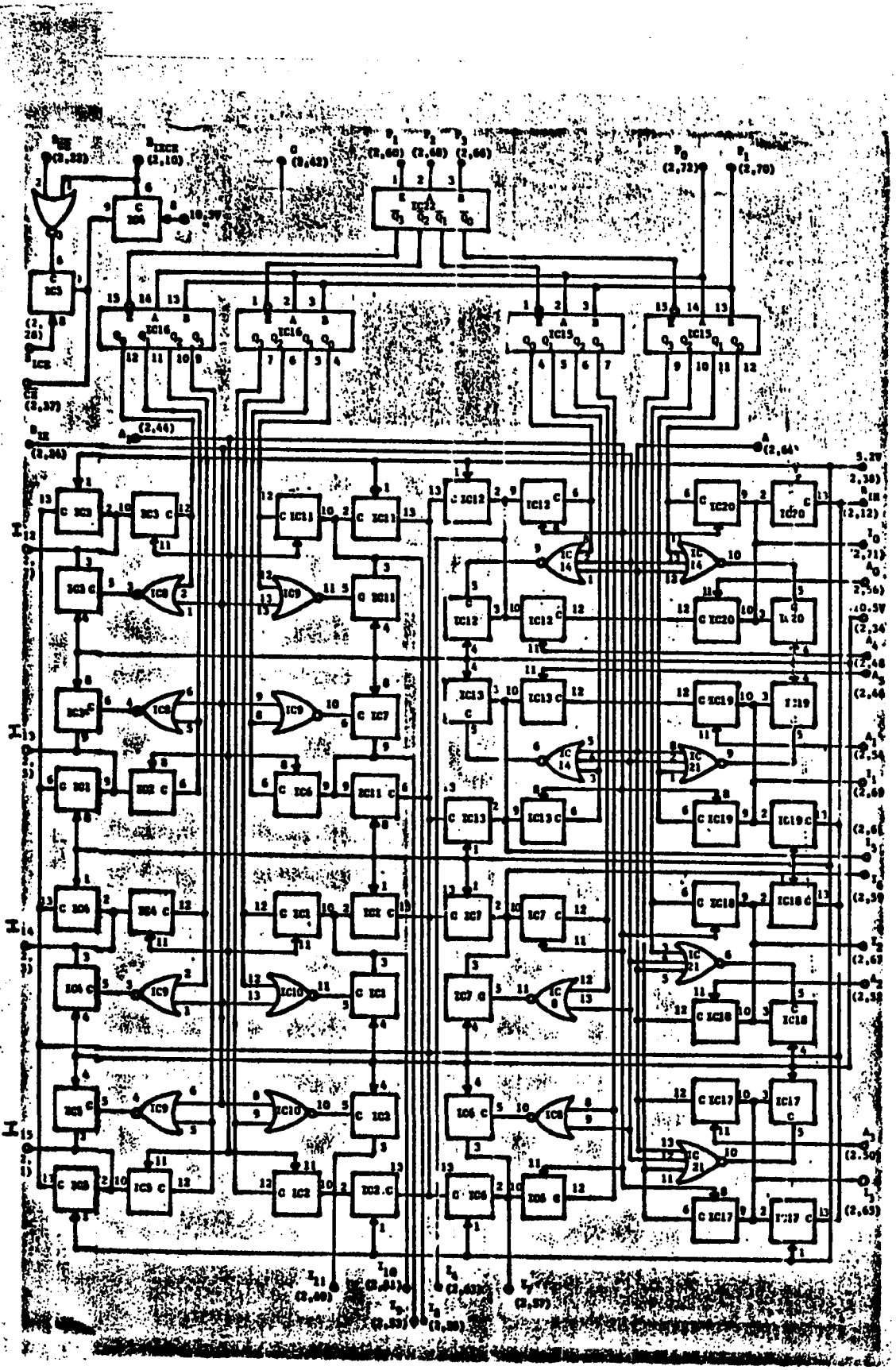


Figure 6.4 Schematic Circuits for Board 2.

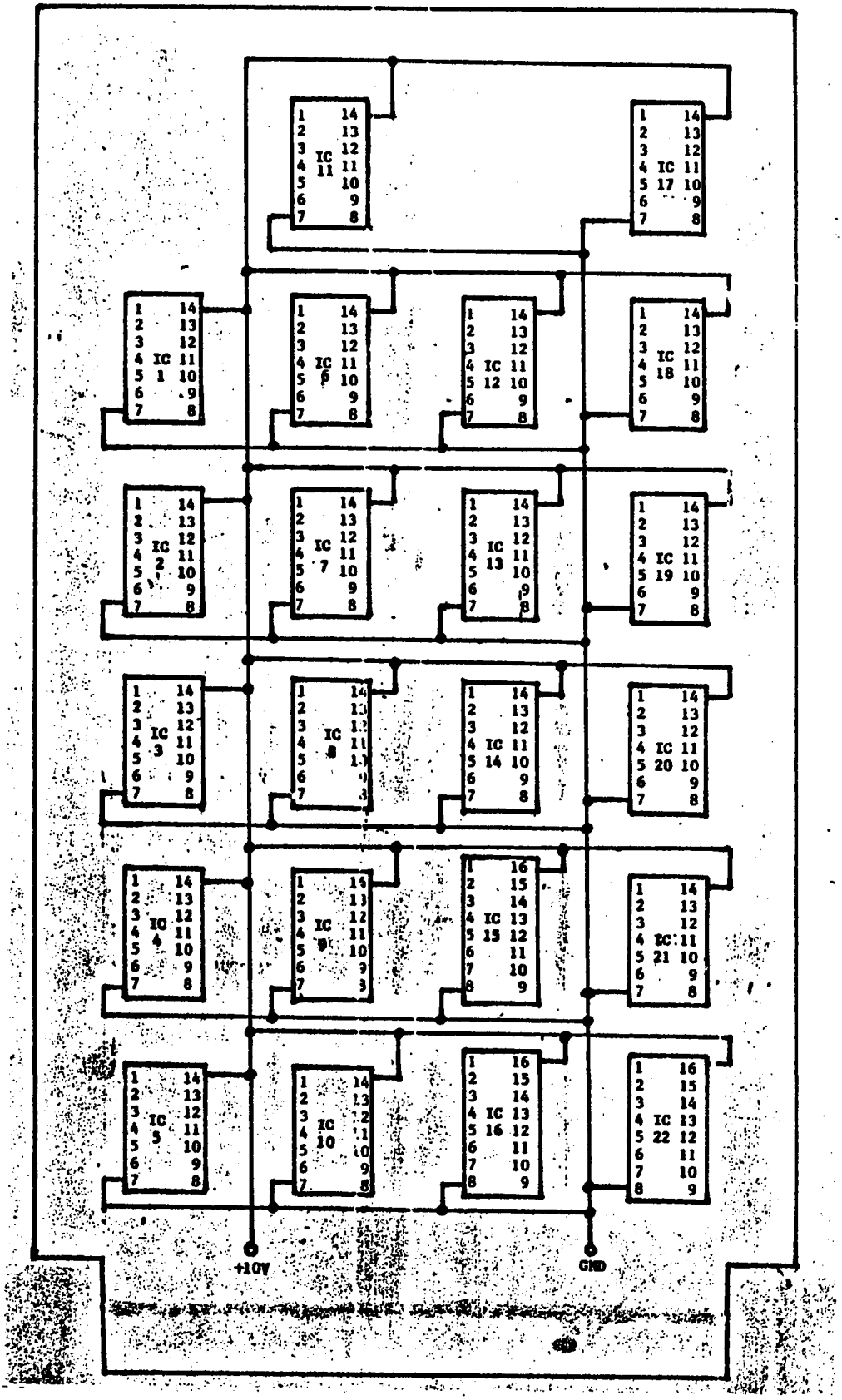


Figure 6.5 IC Organization for Board 2.

		FRONT	BACK	
		1	2	
		3	4	
		5	6	
		7	8	
		9	10	B_{IXCE}
		11	12	B_{IH}
		13	14	
		15	16	
		17	18	
		19	20	
		21	22	$B_{\overline{CE}}$
		23	24	B_{IX}
		25	26	
		27	28	B_{ICE}
		29	30	
		31	32	
		33	34	+10.5V
		35	36	
	$\overline{CE}$	37	38	+5.2V
		39	40	
Input Variable 15.	I_{15}	41	42	GND
	I_{14}	43	44	A_I
	I_{13}	45	46	A_5
	I_{12}	47	48	A_4
	I_{11}	49	50	A_3
	I_{10}	51	52	A_2
	I_9	53	54	A_1
	I_8	55	56	A_0
	I_7	57	58	
	I_6	59	60	P
	I_5	61	62	
	I_4	63	64	A
	I_3	65	66	P_3
	I_2	67	68	P_2
	I_1	69	70	P_1
	I_0	71	72	P_0

(Note. Empty locations : No connection)

Figure 6.6. I/O Signals for Board 2.

TABLE XX Parts List for Board 2

IC #	PART #	DESCRIPTION	QTY
22	F 4556 PC	3-8 decoder (low output)	1
15,16	F 4555 PC	3-8 decoder (high output)	2
1-7, 11-13, 17-20	F 4016 PC	Transmission gate with low r_{on} resistance ($\simeq 50\Omega$)	14
14, 22	F 4025 PC	Tri, 3-input NOR	2
8,9,10	F 4001 PC	Quad, 2-input NOR	3
	-	72-pin socket	1
	-	72-I/O IC board	1

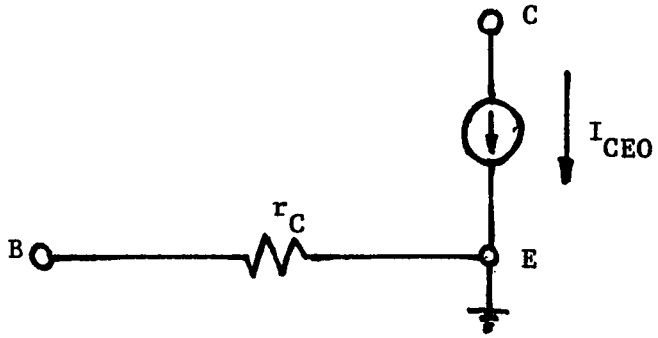
6.2.5 Switching Transistor Analysis

The common emitter transistor operates in three regions:

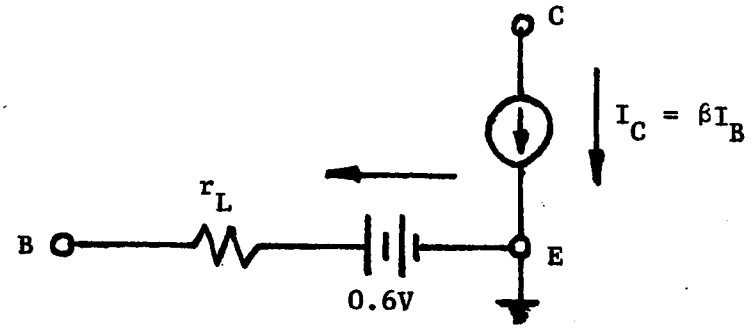
1. Cutoff region; when $V_{BE} < 0.6$ V
2. Active (linear) region; when $0.2 \leq V_{CE} \leq V_{CC}$, $I_C = \beta I_B$.
3. Saturation region; when $V_{CE} \leq 0.2$ V

The transistor equivalent circuits for the three regions are shown in Fig. 6.7. The equivalent circuits are for a NPN transistor. In the cutoff region the input resistance is very high, and the transistor output is nearly an open circuit, with a leakage current less than 1 μa. In the linear region, the input resistance is about 1 K and the collector current follows the base current multiplied by a factor of β. In the saturation region the collector current depends on V_{CC} and the saturation resistance r_s with $V_{CE} \leq 0.2$ V.

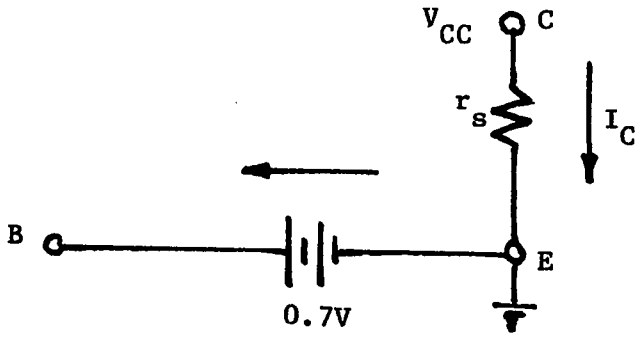
To insure that the transistor gets into the saturation region, a forced gain is used, determined by a divisor factor from 1.5 to 3 of $(h_{FE})_{min}$. The higher the divisor factor, the better the likelihood that the transistor is in saturation. Some of the specifications of the transistors used in PLA interface are shown in Table XXI [21, 22]. The above concepts and specifications will be utilized in the subsequent subsections.



a. Cutoff Region.



b. Linear Region.



c. Saturation Region.

Figure 6.7. Transistor Equivalent Circuits.

TABLE XXI Specifications for Transistors .

#	$(h_{FE})_{min}$	$(IC)_{max}$ (A)	I_{CEO} (MA)	TYPE
2N 3904	40	200×10^{-3}	0.1×10^{-3}	NPN Si
2N 3638A	80	500×10^{-3}	50×10^{-6}	PNP Si
TIP 32A	20	3.0	0.3	PNP Si
TIP 140	500	10.0	2	NPN Si

6.2.6 PLA Supply Voltage Pin (V_{CC})

There are four states for V_{CC} |12|

1. V_{CCL} : $0 \leq V_{CCL} \leq 0.8$ V @ 350 mA
2. V_{CCP} : $4.75 \leq V_{CCP} \leq 5.25$ V @ 1000 mA
3. V_{CCS} : $8.25 \leq V_{CCS} \leq 8.75$ V @ 1000 mA
4. Open : High impedance

Only one of the states shown above can exist at a particular time. Software is utilized to accomplish this task. The circuit shown in Fig. 6.8 is used to realize the required V_{CC} values. When B_{CCP} is high, V_{CC} will be at V_{CCP}. The following equations are used to find values of R_1 and R_2 such that both transistors will saturate.

Transistor Q_2 has $(h_{FE})_{min}$ of 20 (Table XXI). Utilizing the factor of 1.5, $(h_{FE})_{forced}$ is 13.33. Hence:

$$I_{B2} = I_{C1} = I_{C2}/(h_{FE})_{forced} = 1/13.3 = 75 \text{ mA}$$

$$R_2 = \frac{V_E - (V_{BE})_{sat} - (V_{CE})_{sat}}{I_{B2}} = \frac{5.2 - 0.7 - 0.2}{75 \times 10^{-3}} = 57.33\ \Omega$$

Where: V_E is the emitter voltage

$(V_{BE})_{sat}$ is the base-emitter voltage at saturation

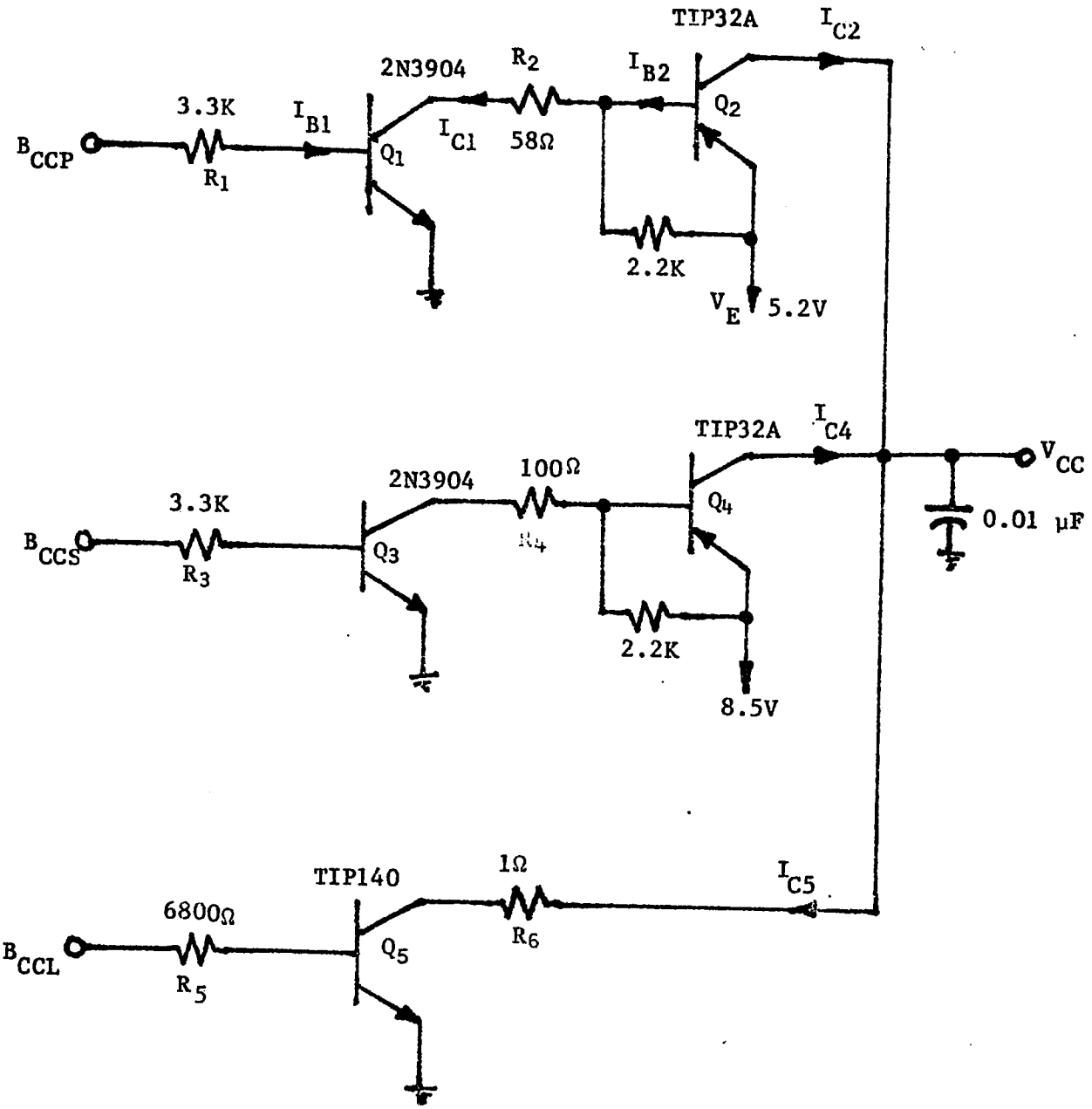


Figure 6.8 Circuit Used to Realize the Supply Voltage States.

$(V_{CE})_{sat}$ is the collector-emitter voltage at saturation.

The value of R_2 = 58Ω was chosen.

Similarly for $(h_{FE})_{min}$, Q_1 = 40 and $(h_{FE})_{forced}$ = 25

$$I_{B1} = I_{C1}/25 = 3.00 \text{ mA}$$

$$R = \frac{V_{OH} - (V_{BE})_{sat}}{I_{B1}} = \frac{10.5 - 0.7}{3.0 \times 10^{-3}} = 3266.67 \ \Omega$$

Where: V_{OH} is the minimum generated CMOS output high voltage.
The value selected is R_1 = 3.3 KΩ.

When BCCS is high, V_{CC} will be at V_{CCS}. Computing similarly so for V_{CCP}, the value of R_3 and R_4 are 3.3 KΩ and 100Ω respectively.

When B_{CCL} is high, V_{CCL} will be applied to the V_{CC} pin.
For this case:

$$(R_6)_{max} = \frac{(V_{CCL})_{max} - (V_{CE})_{sat}}{I_{C5}} = \frac{0.8 - 0.2}{350 \times 10^{-3}} = 1.71\Omega$$

Choosing R_6 = 1 implies $V_{CCL} \leq 0.35$ V.

$$R_5 = \frac{V_{OH} - (V_{BE})_{sat}}{350 \times 10^{-3}/250} = 7 \text{ KΩ}.$$

The selected value is R_5 = 6800 Ω.

The 2.2 KΩ resistors connected to transistors Q_2 and Q_4 are used to decrease the fall time. The 0.01 μF capacitor connected to V_{CC} pin is recommended by the manufacturer.

When V_{CCP}, V_{CCS} and V_{CCL} are low, the V_{CC} pin will be connected to nearly an open circuit. The leakage currents of Q_2 and Q_4 and Q_6 are small. The total leakage current, using Table XXI:

$$|I_{LT}| = |I_{C2L} + I_{C4L} - I_{C5L}|$$

Where: I_{C2L} is the leakage current at I_{C2} when Q_2 is off,
$0 \leq I_{C2L} \leq 0.3$ mA.

I_{C4L} is the leakage current at I_{C4} when Q_4 is off,
$0 \leq I_{C4L} \leq 0.3$ mA

I_{C5L} is the leakage current at I_{C5} when Q_5 is off,
$0 \leq I_{C5L} \leq 2$ mA

Therefore:
$|I_{LT}| \leq 2$ mA.

6.2.7 Fuse Enable Pin (FE)

The following states are to be assumed by pin FE of the PLA |12|:

1. V_{FEL}: $1.25 \leq V_{FEL} \leq 1.75$V @ -1 mA
2. V_{FEH}: $16 \leq V_{FEH} \leq 18$V, -10 $\leq$ Rise time $\leq$ 50 μsec, @ 325 mA
3. Open: High impedance

The circuit shown in Fig. 6.9 is used to realize the FE states. When B_{FEL} is high, V_{FE} will be at V_{FEL}. The analysis equations are similar to the computation in section 6.2.6 and the computed values are R_{12}=1.5 KΩ and R_{11} = 20 KΩ. When $B_{FEH} = B_{FEL} = 1$, Q_7 and Q_8 are ON. With V_{FE} at 17.5V, the current passing through the 1.5 KΩ resistor will be:

$$I_{R_{12}} = \frac{17.5 - (V_{CE})_{sat7} - (V_{CE})_{sat8}}{R_{12}} = \frac{17.5 - 0.2 - 0.2}{1.5 \times 10^3} = 11.6 \text{ mA}$$

Transistor Q_8 will tolerate a maximum collector current of 200 mA, while Q_7 can tolerate 500 mA (Table XXI). Therefore no damage is inflicted on the circuit for this input condition. The output voltage V_{FE} will be at about 17.3 V (17.5 - $(V_{CE})_{sat}$) which is in the limits of V_{FEH}.

When V_{FE} is to have a transition to the value V_{FEH}, its rise time (T_R) is to be about 30 μsec ($10 \leq T_P \leq 50$ μsec). During most of this rise time interval the collector current of Q_7 essentially follows the base current by a linear relationship. Thus, if the base current is adjusted such that it has a rise time of 30 μsec in the linear region and transistor Q_7 does not saturate, then the desired output is achieved. The equivalent circuit for Q_6, which provides the base current, is drawn in Fig. 6.10. Since B_{FEH} is high during the rise time, the diode is reverse biased. Part (a) of Fig. 6.10 represents Q_6 in its cutoff region, while part (b) is for Q_6 in the linear region. Transistor Q_6 has a very high input resistance compared to R_9 because of a small base to emitter voltage. Hence, the parallel combinations of r_{off}, R and r_C, R_9 are nearly

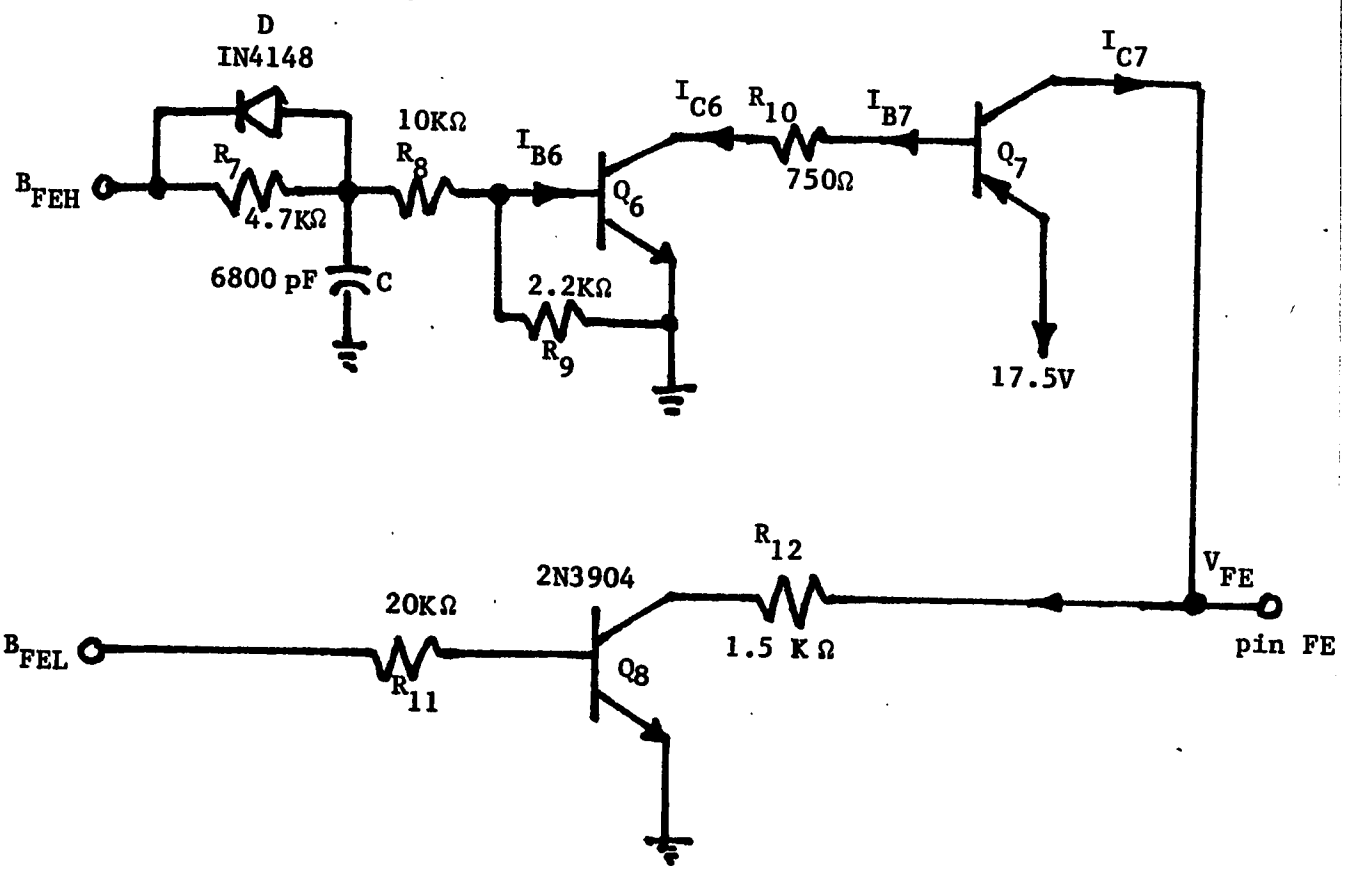
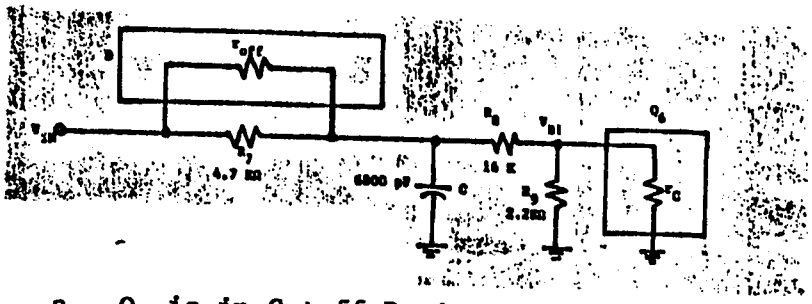
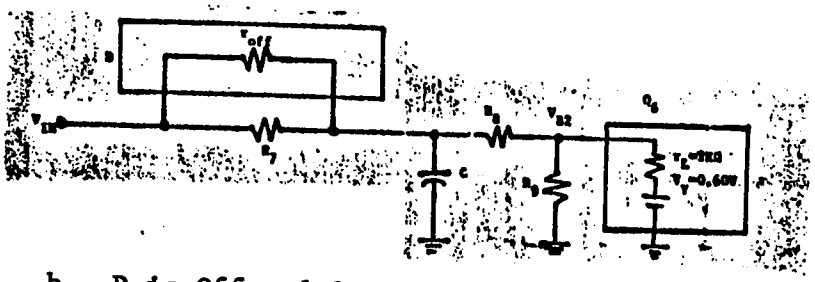


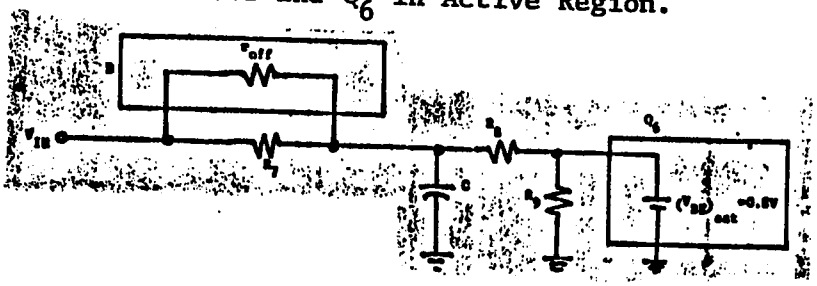
Figure 6.9 Circuit Diagram for Fuse Enable Pin (FE).



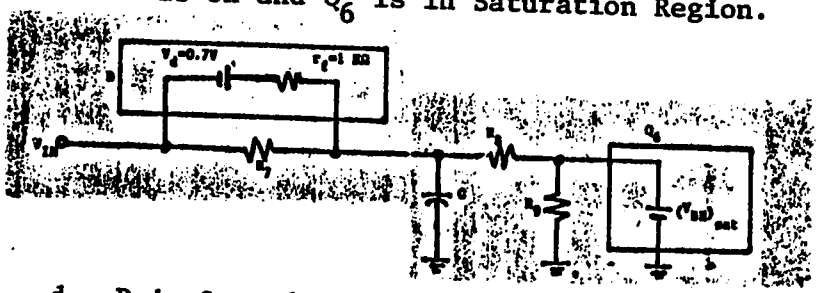
a. Q_6 is in Cutoff Region and D is Off.



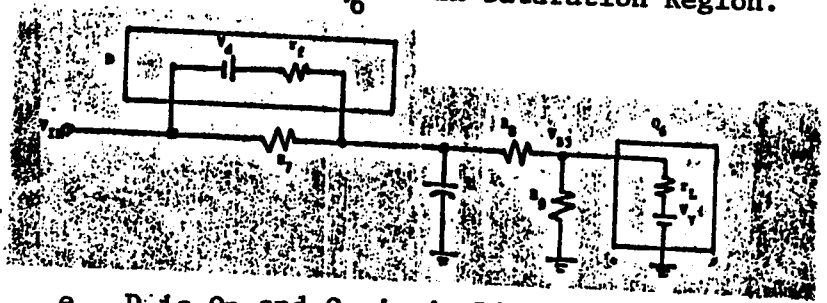
b. D is Off and Q_6 in Active Region.



c. D is on and Q_6 is in Saturation Region.



d. D is On and Q_6 is in Saturation Region.



e. D is On and Q_6 is in Linear Region.

Figure 6.10 Circuits Used to Calculate Rise Time and Fall Time of V_{FE}.

R_7 and R_9, respectively. The voltage V_{B1} (Fig. 6.10a) will therefore have a maximum steady state voltage (V_{BS}) of:

$$V_{BS1} = \frac{V \quad R_9}{R_7 + R_8 + R_9}$$

Where: V is the steady state voltage of V_{IN}.

The time constant for the circuit is:

$$\tau_1 = \frac{R_7(R_8 + R_9)}{R_7 + R_8 + R_9} C$$

Assuming zero initial conditions,

$$V_{B1} = V_{BS1}(1 - e^{-t/\tau_1}) = 1.01(1 - e^{\frac{-t}{25.4x10^{-6}}})V$$

When V_{B1} is at about 0.60 V, Q_6 will enter the active region. In the linear region r_L is about 2.0 KΩ and V_γ = 0.60 V, as shown in Fig. 6.10b. The Thevenin equivalent for r_L, V_γ and R_9 is V'_γ = 0.31 and R'_9 = 1047.6 Ω. The steady state value for V_{B2} (V_{BS2}) is:

$$V_{BS2} = \frac{(V - V'_\gamma)R'_9}{R_7 + R_8 + R_9} + V'_\gamma$$

The time constant for the circuit is:

$$\tau_2 = \frac{(R'_9 + R_8)R_7}{R_7 + R_8 + R_9} C$$

The initial condition for V_{B2} is 0.65 when the base input of Q_6 has a change in its equivalent circuit.

Therefore:

$$V_{B2} = (V_{BS2} - 0.60)(1 - e^{-t/\tau_2}) + 0.60V$$

$$= 0.2(1 - e^{\frac{-t}{25x10^{-6}}}) + 0.60V \qquad (6.2)$$

Transistor Q_7 base current corresponds to the collector current.

The minimum I_{C6} required to saturate Q_7 in 4.4 mA ($= I_{C7}/(h_{FE7})_{min} = 355x10^{-3}/80$). Since Q_6 is in the linear region:

$$I_{C6} = \beta I_{B6}$$

Where: β is the current gain ($\simeq 55$).

Therefore for I_{CL} of 4.4 mA:

$$I_{B6} = \frac{4.4x10^{-3}}{55} = 0.08 \text{ mA}$$

At $I_{B6} = 0.04$, $V_{B2} = I_{B6} \times r_L + V_\gamma = 4.4x10^{-5}x2 \times 10^3+0.60 = 0.76V$. Since the PLA represents a resistive load, the rise time (T_R) for V_{FE} is the rise time of I_{C7}. Furthermore, the rise time of I_{C7} is approximately the rise time of I_{C6} from I_{CE06} to $(I_{C6})_{min}$. Again because of linearity, the rise of I_{C6} between the aforementioned values is the rise time of V_{B2} from 0.60V to 0.76V.

Rise time is defined to be the time required for a voltage transition from $[0.1(V_2-V_1)+V_1]$ to $[0.9(V_2-V_1)+V_1]$.

Where: V_2 is the final voltage (0.76V),

V_1 is the initial voltage (0.6).

Using equation 6.2 to calculate rise time:

at $V_{B2} = 0.60V$, $t_1 = 2.08$ µsec

at $V_{B2} = 0.76V$, $t_2 = 31.82$ µsec

Therefore $T_R = t_2 - t_1 = 29.74$ µsec. This value is within the rise time limits specified by the manufacturer ($10 \leq T_R \leq 50$ µsec). As mentioned earlier Q_7 will go into saturation before Q_6. The collector current of Q_7 at saturation will obey:

$$I_{C6} = \frac{V_{CC}}{r_S + R_L}$$

Where: R_L is the load resistance computed from the manufacture's specification of 355 mA at about 17.5V,

r_s is the saturation resistance for the saturation current at Q_7 ($V_{CE} \simeq 0.2V$)

The circuits in Fig. 6.10d and e are used to calculate the fall time. The diode has a forward resistance r_f of about 1 K and V_d is about 0.7 V. Hence the Thevenin equivalent circuit for the diode and R_7 is $V_d' = 0.58V$ and $R_7' = 824.56$. At the beginning, Q_6 is in saturation with its V_{BE} about 0.8V (Fig. 6.10d) and will remain there until I_{B6} gets below $I_{B6\ min}$ of 0.55 mA $(\frac{V_{E7} - (V_{BE7})_{sat} - (V_{CE6})_{sat}}{(h_{FE6})_{min}})$. When Q_6 enters the linear region, the circuit in Fig. 6.10e is used. Transistor Q_7 will remain in saturation until I_{C6} goes below $I_{B7_{min}}$ (4.4 mA). For I_{B7} of 4.4 mA and using β of 55, I_{B6} will be at 0.08 mA and V_{B3} at 0.76V.

Transistor Q_6 base voltage is (using Fig. 6.10e):

$$V_{B3} = \left(\left(\frac{V_{IN} - V_\gamma' + V_d'}{R_7' + R_8 + R_9}\right)R_9' + V_\gamma' - V_{Bi}\right)\left(1 - e^{-t/\tau_3}\right) + V_{Bi}$$

Where:

V_{Bi} = initial V_{B3} voltage when Q_6 goes into the linear region from saturation $\simeq 0.8V$

$$\tau_3 = \frac{(R_8 + R_9') + R_7'}{R_7' + R_8 + R_9'} e = 5.3\ \mu sec.$$

Therefore:

$$V_{B3} = -0.47\left(1 - e^{\frac{-t}{5.3x10^{-6}}}\right) + 0.8\ V.$$

The fall time is the time required for V_{B3} to fall from

0.76 V, at which I_{B6} is about 0.08 mA, to about 0.6 V at which Q_6 will enter cut-off (actually Q_7 will enter cut-off prior to $V_{B3} = 0.6$ V because I_{C6} will be very small ($\simeq I_{CE06}$) before that value.):

at $V_{B3} = 0.62$, $t_2 = 2.56$ μsec

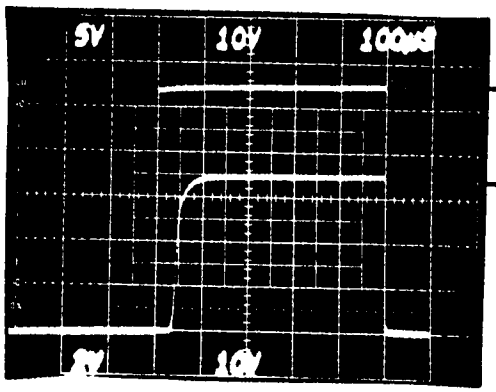
at $V_{BS} = 0.74$, $t_1 = 0.72$ μsec.

Therefore the fall time is about 1.84 μsec.

The input/out signals for the circuit of Fig. 6.10 are shown in Fig. 6.11a. Both signals have a common abscissa. The reader should notice the negligible fall time and the 30 μsec rise time, expanded in Fig 6.11b. The reader should bear in mind that the transistor parameters differ from one transistor to another of similar type. The analysis of the circuit followed in this section will be different for another pair of transistors; Q_6 and Q_7. Hence rise time will vary from one pair to another and so it is suggested, for better controlled output characteristics, that variable resistances to be used for R_8 and R_9 and they should be adjusted each time Q_6 and Q_7 are changed.

The reader should also notice that in the analysis the capacitance of the devices were neglected. Because the transistors and the diode are switching devices, low capacitances exist.

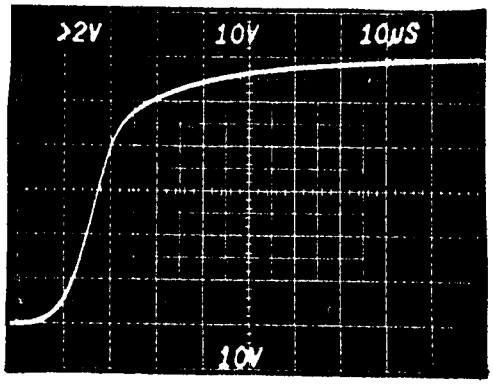
When both B_{FEL} and B_{FEH} are low, V_{FE} will be nearly open with a leakage current of about 100 mA. The leakage current is the algebraic sum of the two leakage currents resulting from Q_7 and Q_8.



V_{IN} at 2V

V_{OUT} at 5V
(Same Abcissa)

a. I/O Signals for V_{FEH}



V_{OUT} @ > 2V

b. Rise Time of V_{FEH} about 30 µsec.

Figure 6.11 V_{FEH} I/O Signals and Rise Time Waveforms.

6.2.8 Circuit for Output Functions Pins

Each output function assumes one of the following states [12]:

1. V_{OPH}: $16 \leq V_{OPH} \leq 18$ V @ 325 ma.
2. V_{OPL}: $0 \leq V_{OPL} \leq 0.8$ V @ -1 ma.
3. V_{OPF}: $9.5 \leq V_{OPF} \leq 10.5$ V @ 10 ma.
4. Open: High impedance

Also F_0 through F_5 have two more possible states:

5. V_{OHF}: $2.4 \leq V_{OHF} \leq 5.5$ V @ 100 μA
6. V_{OLF}: $0 \leq V_{OLF} \leq 0.8$ V @ -1 mA

The circuit for the output F_J is shown in Fig. 6.12. The generation of states V_{OPH} and V_{OPL} is similar to that used for pin FE (V_{FEH} and V_{FEL}), except that V_{OPH} is selected via a 4-10 decoder. When B_{OPF} is high, the output selected by the 4-10 decoder is set to V_{OPF}. The resistance of the TG is 31Ω (Table XVIII), hence using equation 6.1 :

$$V_{FJ} = 10.5 - 10 \times 10^{-3} \times 31 = 10.2 \text{ V}$$

For address signals, when A_F is high, F_0 through F_5 will assume the same values as the inputs A_0 through A_5. When A_J is low

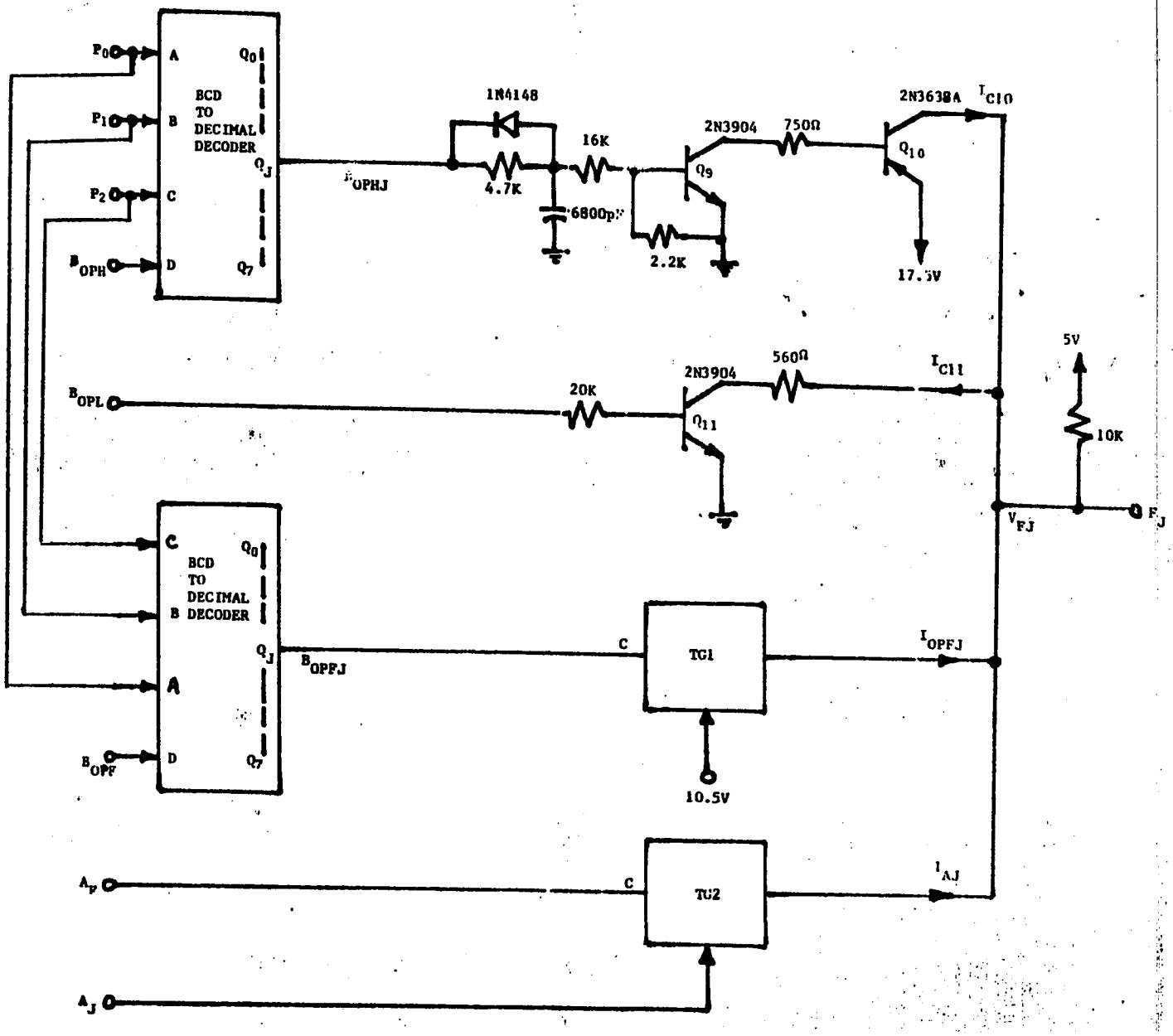


Figure 6.12 Circuit Diagram for Output Functions.

V_{FJ} will be at:

$$V_{FJ} = 0.5 - (-1 \times 10^{-3}) \times 29 = 0.53 \text{ V}$$

If A_J is high (5 V), using equation (6.1) and Table XVIII, V_{FJ} will be

$$V_{FJ} \simeq 5.2 - 100 \times 10^{-6} \times 29 = 5.2 \text{ V}$$

When all circuits are open for a given output function pin, V_{FJ} will be nearly open with a leakage current of:

$$\left|I_{FJL}\right| = \left|I_{C10L} + I_{OPFJL} + I_{AJL} - I_{C11L}\right|$$

The total leakage current is approximately 1 μA.

6.2.9 Sensing Function

Each output function pin is connected to a TG. In turn all outputs of the TG's are connected through a common point to the CMOS-TTL interface. The final result is transmitted back to the microcomputer via a tri-state buffer, as shown in Fig. 6.13. The output to be sensed is selected via a 4-10 decoder, with bit S_J used as the TG enable (C). Output 7 can be selected via the decoder or via bit "Sense 7". This is because, in verifying the AND matrix, bits P_0 through P_3 will be representing the P-term to be verified. Hence, the decoder cannot be used to select output 7. A resistor to ground is also connected to the input of the 10V/5V

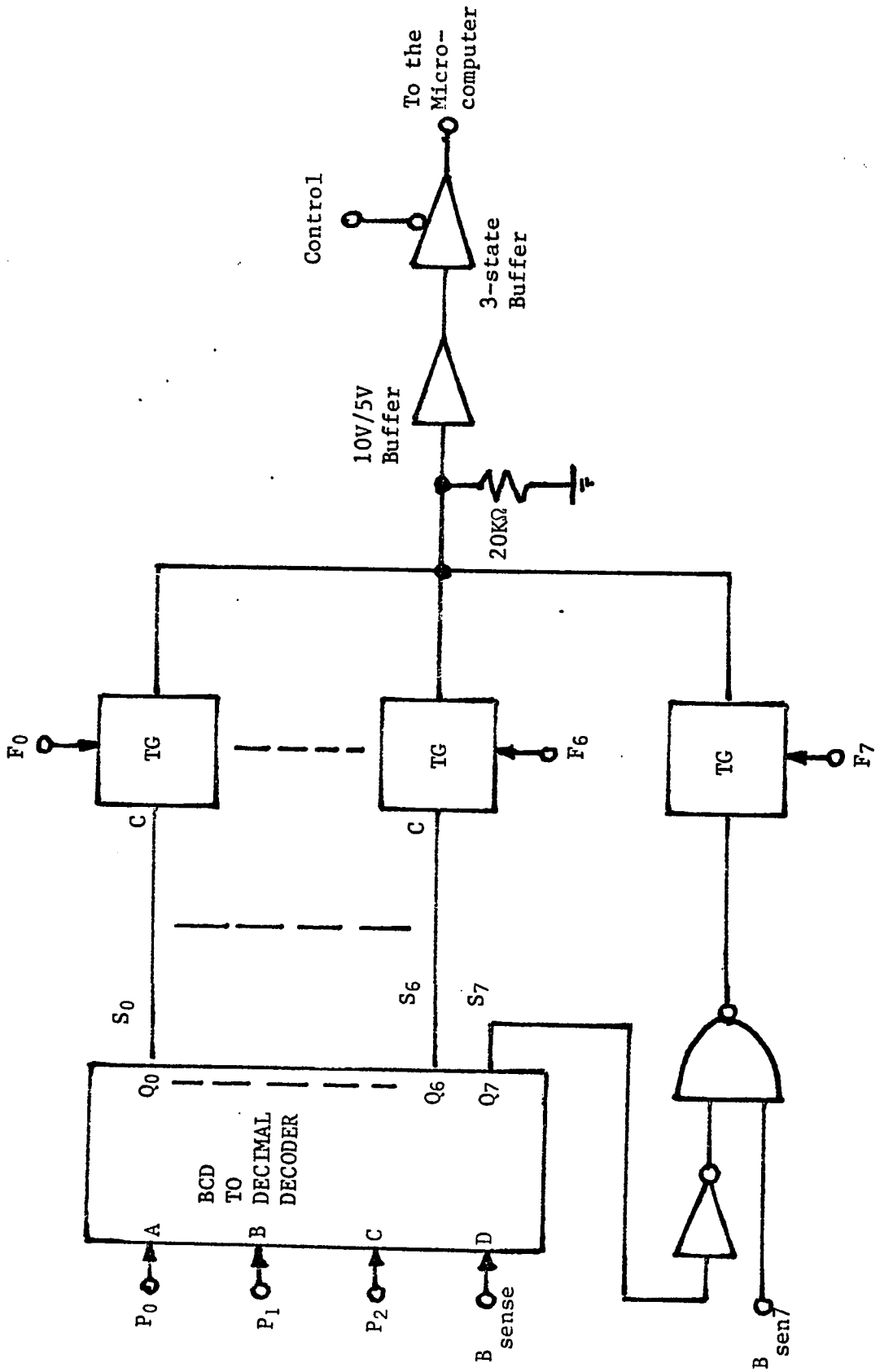


Figure 6.13 "Sensing" Circuit Diagram.

buffer to prevent an open circuit, when all the TG's are open.

6.2.10 Circuit Assembly for PLA, V_{CC}, FE, and Output Pins

The supply voltage, fuse enable, and output functions circuits are assembled on one board (board 3). The schematic diagram is shown in Fig. 6.14 and the IC layout is shown in Fig. 6.15. The parts list is provided in Table XXII. The I/O signals to the board are shown in Fig. 6.16.

6.3 COMPARISON OF CIRCUIT SIGNALS AND THOSE SUPPLIED BY THE MANUFACTURER

Photographs were taken for the three burn-in procedures of signals so as to draw comparisons with the diagrams supplied by the manufacturer [12]. Each photograph shows pertinent voltage signals and the PLA pins for which these signals apply. Figure 6.17 shows waveforms of the output burn-in procedure. In this photograph, outputs F_0, and F_1 are chosen to show T_{PR} and T_{PS} and V_{OPH}, V_{OPL}, t_p. The programming pulse V_{OPH} is to be about 17 volts, with a rise time (T_R) of about 30 μsec and a pulse width (t_p) of 0.5 msec. In the photo V_{OPH} is 7.5 volts, the rise time is small and cannot be seen on the scale chosen, t_p is 0.5 msec.

When verifying the respective output V_{CC} should be increased from V_{CCL} of 0 volts to V_{CCS} of 8.5 volts. Also $\overline{CE}$ drop from V_{IH}

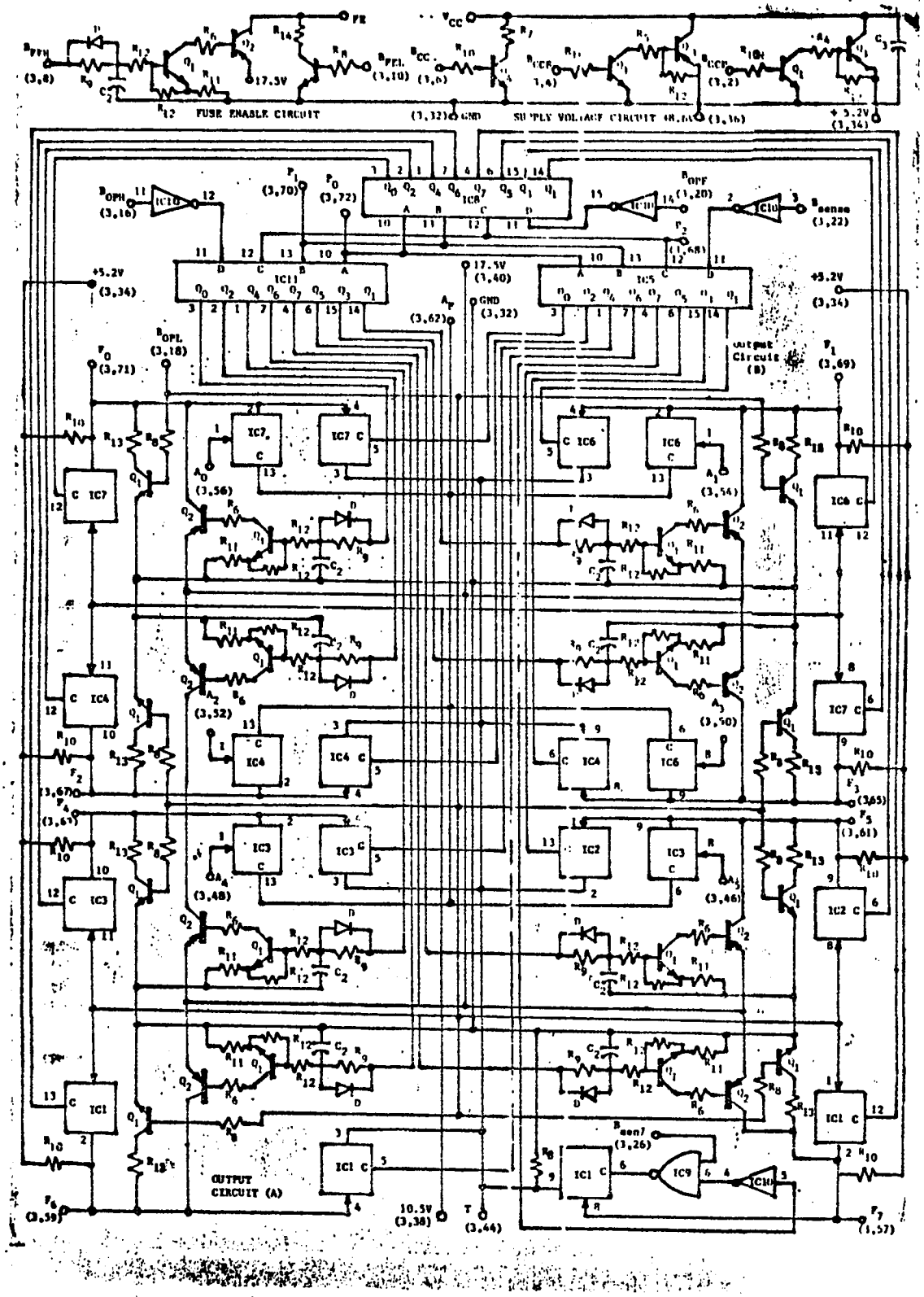


Figure 6.14 Circuit Diagram for Board 3.

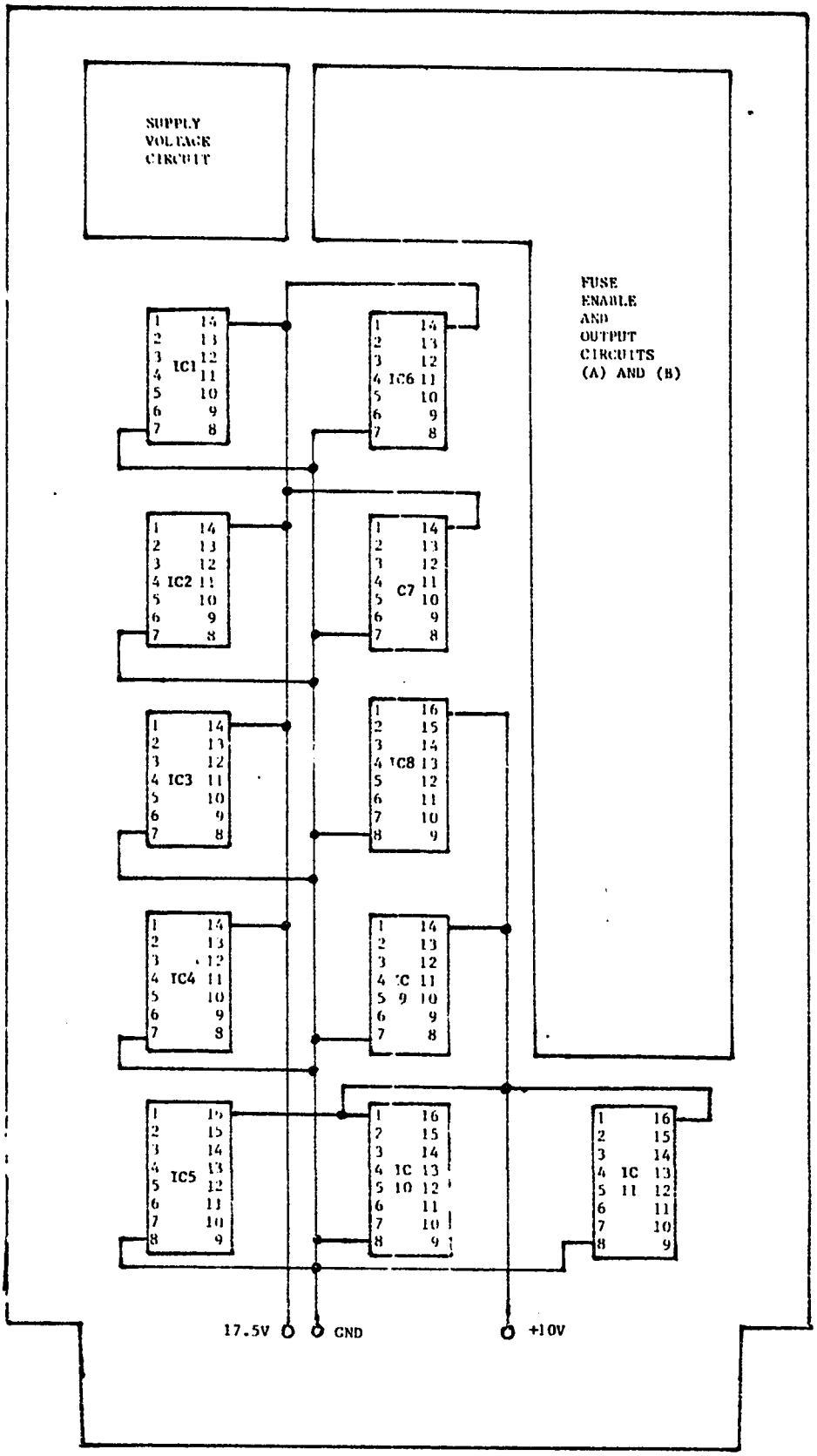


Figure 6.15 IC Organization for Board 3.

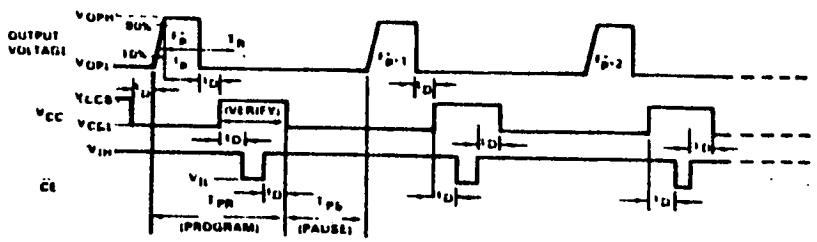
		FRONT	BACK	
		1	2	B_{CCP}
		3	4	B_{CCS}
		5	6	B_{CCL}
		7	8	B_{FEH}
		9	10	B_{FEL}
		11	12	
		13	14	
		15	16	B_{OPH}
		17	18	B_{OPL}
		19	20	B_{OPF}
		21	22	B_{sense}
		23	24	
		25	26	
		27	28	B_{sen7}
		29	30	
		31	32	GND
		33	34	+5.2V
		35	36	+8.6V
		37	38	+10.5V
		39	40	+17.5 V
		41	42	
		43	44	T Information from PLA
		45	46	A_5 Interface
		47	48	A_4
Supply Voltage	V_{CC}	49	50	A_3
Fuse Enable	FE^{CC}	51	52	A_2
		53	54	A_1
		55	56	A_0
Output Function 7	F_7	57	58	
	F_6	59	60	A_F
	F_5	61	62	
	F_4	63	64	
	F_3	65	66	
	F_2	67	68	P_2
	F_1	69	70	P_1
	F_0	71	72	P_0

(Note. Empty locations : No connection).

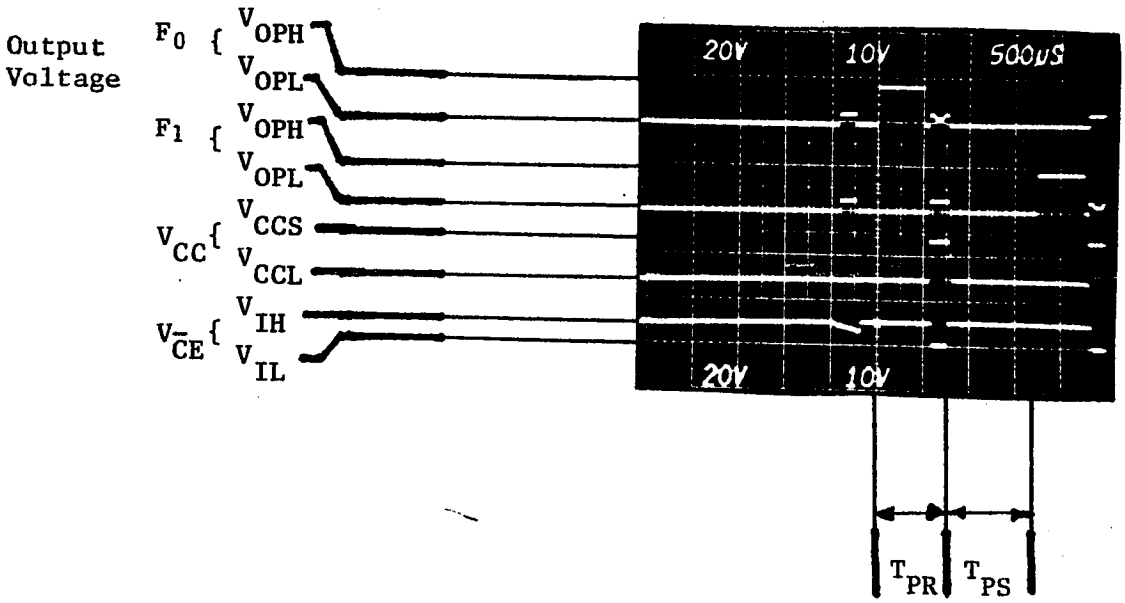
Figure 6.16 I/O Signals for Board 3.

TABLE XXII Parts List for Board 3 .

#	PART #	DESCRIPTION	QTY
IC1-4,6,7	F 4016 PC	Transmission gates, low r_{on}	6
IC 5,8,11	F 4028 PC	BCD-to decimal decorder	3
IC 9	F 4011 PC	Quad 2-input NAND gate	1
IC 10	F 4049	Hex inverter	1
D	1N 4148	Switching diode	8
Q_1	2N 3904	Switching transistor with IC_{max}=200mA	20
Q_2	2N 3638 A	Switching transistor with IC_{max}=1A	9
Q_3	TIP32A	Switching Transistor with IC_{max}=3A	2
Q_4	TIP140	Switching Transistor with IC_{max}=10A	1
R_4		75 Ω resistor, 1/4 watt	1
R_5		110 Ω resistor, 1/4 watt	1
R_6		68 Ω resistor, 1/4 watt	9
R_7		1 Ω resistor, 1/4 watt	1
R_8		20 KΩ resistor, 1/4 watt	3
R_9		8.2 KΩ resistor, 1/4 watt	9
R_{10}		10 KΩ resistor, 1/4 watt	11
R_{11}		680 Ω resistor, 1/4 watt	9
R_{12}		2.2 KΩ resistor, 1/4 watt	20
R_{13}		560 Ω resistor, 1/4 watt	8
R_{14}		1.5 KΩ resistor, 1/4 watt	1
C_2		6800 pF capacitor, 50 V	9
C_3		0.01 μF capacitor, 50 V	1
		72-Pin socket	1
		72 I/O-IC Board	1



a. Signals Supplied by the Manufacturer.



b. Output Signals of the Program.

Figure 6.17 Output Polarity Burn-in and Verify Signals.

(5 V) to V_{IL} of 0 volts, with a delay of t_D. The delay should be greater than 10 μsec. In the circuit it is about 24 μsec. The program time T_{PR} should obey the following eq. |12|:

$$\frac{T_{PR}}{T_{PR} + T_{PS}} < 50\ \%$$

Where: T_{PS} is the pause time.

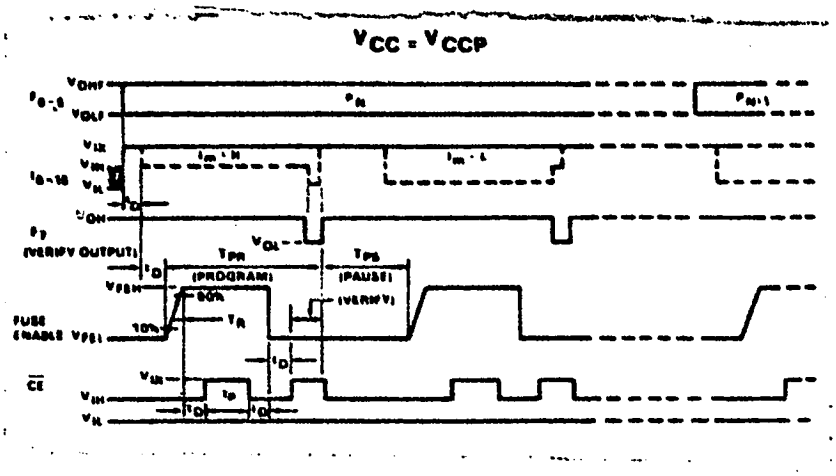
From the photo, T_{PR} = 800 μsec; and $T_{PR} + T_{PS}$ = 1750 μsec.

Therefore:

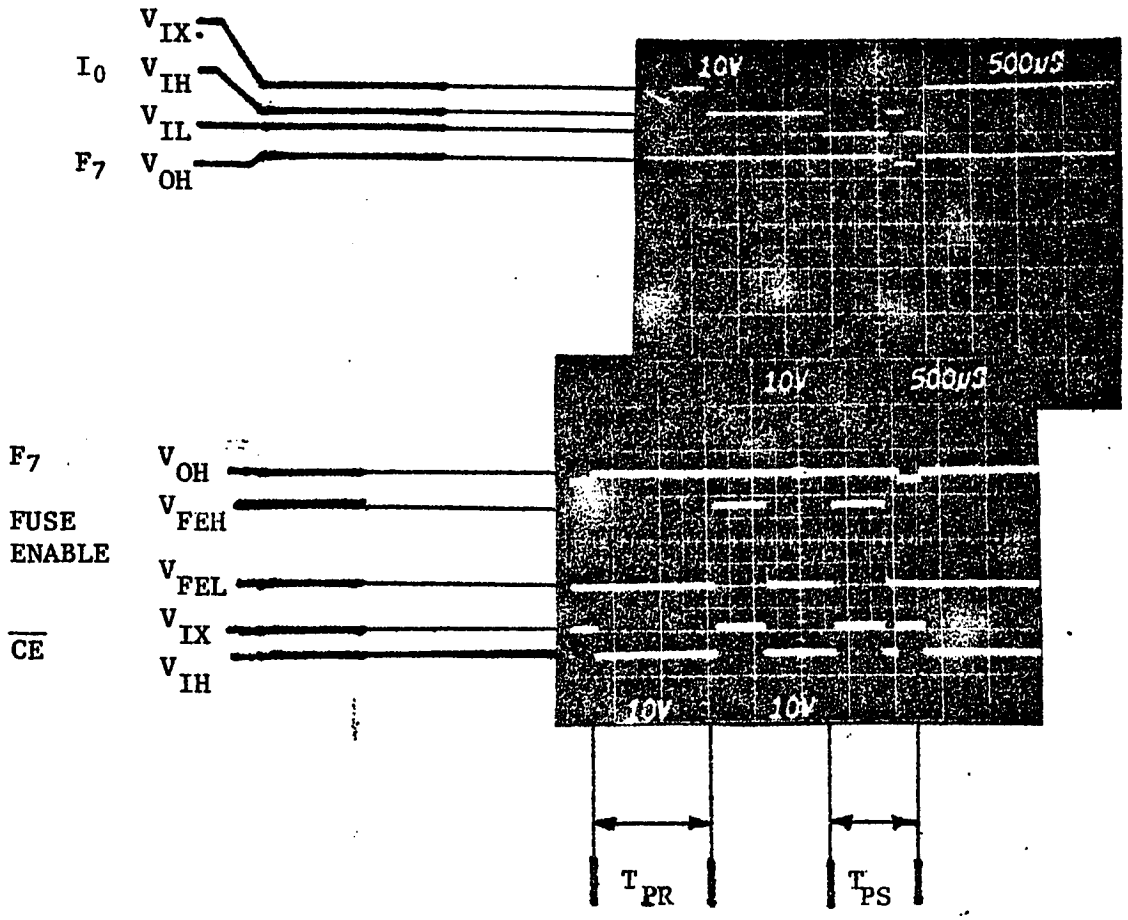
$$\frac{T_{PR}}{T_{PR} + T_{PS}} = 46\ \%$$

Figure 6.18 shows the waveforms of the product burn-in procedure. The quantities F_0 through F_5 are used for addressing the P-term and they are not shown in the photograph.

The OR matrix burn-in procedure waveforms are shown in Figure 6.19. Supply voltage V_{CC} is always set at V_{CCS} (8.5 V) in this procedure, while I_0 through I_5 are used to address the P-term to be used. It is believed that the information provided about the output burn-in can be utilized in understanding the waveforms of Figures 6.18 and 6.19.

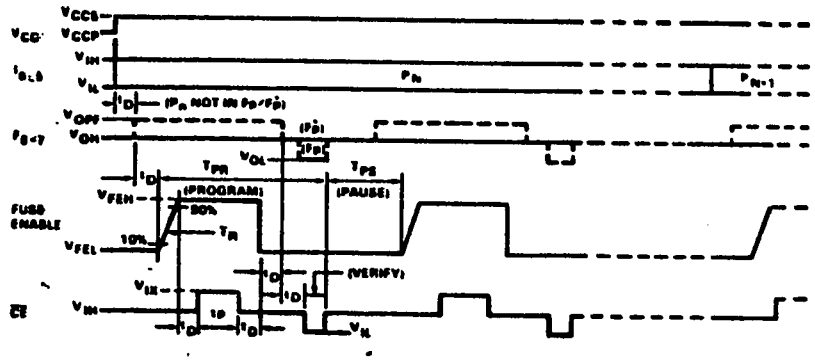


a. Signals Supplied by the Manufacturer

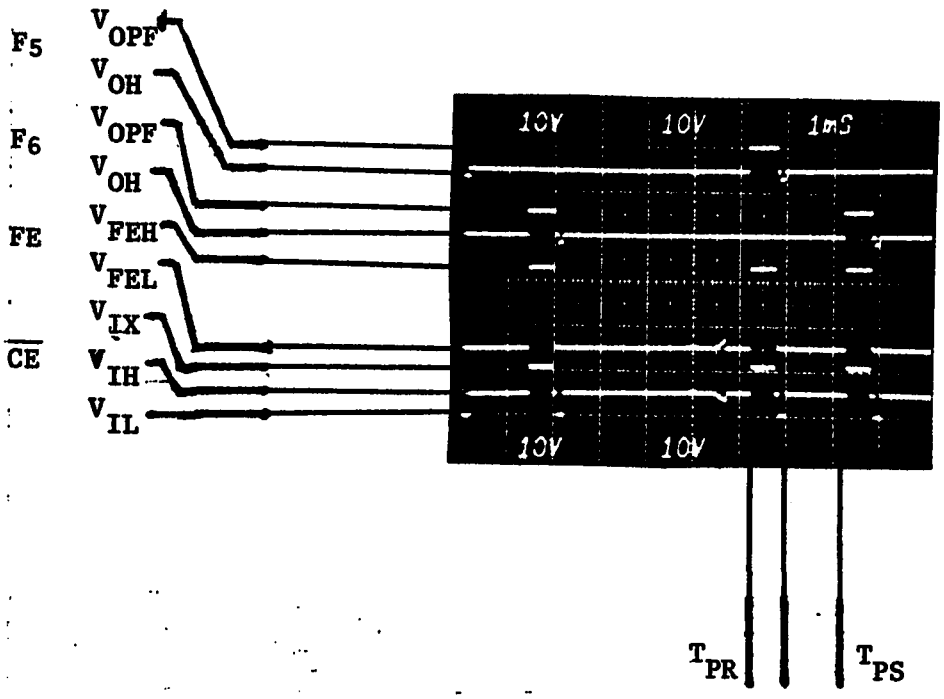


b. Signals As Supplied by "PROD"

Figure 6.18 AND Matrix Burn-in and Verify Signals.



a. Wafeforms Supplied by the Manufacturer.



b. Output Signals of the Program

Figure 6.19 OR Matrix Burn-in and Verify Signals.

VII. SUMMARY

Logical elements are used to fabricate logical designs. To minimize cost, a minimum number of elements should be used. Large scale integration (LSI) makes it possible to have hundreds of gates on only one chip. The LSI technology has reduced digital system cost by this means.

The programmable logic array (PLA) is an LSI device. It is a two level AND-OR logic circuit. A typical PLA has 16 inputs and 8 outputs. It can perform a large variety of digital functions by means of programming the corresponding AND-OR interconnections. A major limitation of its function repertoire is the number of available AND gates. Hence, for economic use of the PLA, a minimization algorithm to reduce the number of AND gates is needed. Furthermore, automatic programming requires that a microcomputer control system be used. The purpose of the computer is to perform the steps necessary to program the PLA. The microcomputer system will minimize time required to program the PLA, as well minimizing ing cost of labor and human errors.

In this work, a prototype microcomputer control system to program the PLA was developed. The minimization algorithm was previously developed and it is operational on the UPM IBM system.

The microcomputer system input data uses hexadecimal notation and the user can use the output of the minimization algorithm as input data. The system operation is principally software dependent.

This was an intentional design strategy to minimize hardware design. The output of the microcomputer controls the interface hardware. At the time being, 29 bits are needed to control the hardware. The system is made flexible for further modification so that verifying and programming procedures can be altered to suit the user's desire. The system, as it now exists, follows the Signetics 82S100/101 PLA specifications for programming a 16 input, 8 output device, with a verifying procedure |12|.

VIII. FUTURE WORK

The hardware and the software developed is capable of programming and verifying the Signetics 82S100/101 PLA. It is important to consider modifications, to the software, as well as for the hardware, and data input.

An interface between the IBM minimization program and the hardware developed can be used through an acoustic coupler or any other means capable of transferring data directly from the IBM system to the programming circuit. Such a communication link is needed to be established.

The software at the time being is designed to verify burned links and type error messages if errors are encountered. It can be modified to stop after an error discovery and print the message, then wait for user instructions. Or it can be programmed to burn other fuses in accordance with user specifications.

As mentioned before, the hardware used consisted of available parts in the UPM laboratory. The hardware can be minimized by using CMOS-TTL interface chips, addressable latches, 4-16 decoder, and 3-8 decoders. The use of PLA's to realize some of the above-mentioned functions is also possible. This will reduce the bits required to control the hardware, as well as decrease the number of chips needed.

One major advantage of the system is that it is principally software dependent. The system is made flexible so that it can be

modified to program devices other than the PLA 82S100/101. The following devices can be programmed using the system with some software modifications |12|:

1. Mask programmable logic array 82S200/201
2. Field programmable ROM patch 82S106/107
3. 8192 - Bit Bipolar RROM 82S180/181

The system is capable to handle a device with 16 inputs and 8 outputs. The device programming and verifying characteristics should be similar to those of PLA 82S100/101 (Appendix A). The number of outputs can be increased to 15 with some modification to the hardware (adding more output circuits).

REFERENCES

|1| Cavlan, Napoleane, "Signetics Field Programmable Logic Arrays", Advanced Product Marketing, Signetics, Sunnyvale, Calif., July, 1975.

|2| Kobylarz, J. and Najjar, A., "The UPM Programmable Logic Array Burn-in Facility", Proceedings of the 5th National Computer Conference, Dhahran, Saudi Arabia, March 20-22, 1979.

|3| "MEMORY DATA BOOK", National Semiconductor, 1977.

|4| Birkner, John, "Microprogramming Random Logic", Proceedings of IEEE 1978 Compcon Spring, pp. 75-80.

|5| Stout and Kaufman, "HANDBOOK OF MICROCIRCUIT DESIGN AND APPLICATION", McGraw Hill Inc., 1980.

|6| Heat, Chris and Williamson, Ian, "Formal Logic Synthesis for the Programmable Logic Array", Electronic Engineering, September, 1976.

|7| Maggiere, Joe, "PLA-a Universal Logic Element", Electronic Product Magazine, April 15, 1974.

|8| Reyling, G., "PLAs Enhance Digital Processor Speed and Cut Component Count", Electronics, August 8, 1974.

|9| Cavalon, Napoleone, "Structure and Applications of Field Programmable Logic Arrays", Advanced Product Marketing Segnetics, Sunnvale, Calif. 1975.

|10| Al-Najjar, A.J.,"MINIMIZATION OF SWITCHING FUNCTIONS AS APPLIED TO PROGRAMMABLE LOGIC ARRAYS", Master Thesis, Department of Electrical Engineering, UPM, March, 1979.

|11| Edwards, C.R., "A New Approach to PROM/FPLA Programming", Electronic Equipment News (GB), April, 1978.

|12| "SIGNETICS BIPOLAR AND MOS MEMORY", Data Manual, Signetics, 1979.

|13| "PACE USERS MANUAL", National Semiconductor, 1977.

|14| "PACE MICROPROCESSOR ASSEMBLY LANGUAGE MANUAL", National Semiconductor, 1977.

|15| "PACE SYSTEM DESIGN MANUAL", National Semiconductor, 1977.

|16| "THE TTL DATA BOOK FOR DESIGN ENGINEERS", Texas Instruments Inc., 1973.

|17| Herserman, David L., "HANDBOOK OF DIGITAL IC APPLICATIONS", Prentice-Hall Inc., 1980.

|18| "COS/MOS INTEGRATED CIRCUITS", RCA 1978.

|19| Gloriose and Streater, "CMOS DESIGNERS' PRIMER AND HANDBOOK", E & L Instruments, Inc., 1978.

|20| Millman-Halkias, "INTEGRATED ELECTRONICS", McGraw-Hill Inc. 1972.

|21| "TRANSISTOR D.A.T.A. BOOK", D.A.T.A. Inc., 1976.

|22| Kaufman and Seidman, "HANDBOOK OF ELECTRONICS CALCULATIONS" McGraw-Hill, 1979.

|23| Fletcher, "AN ENGINEERING APPROACH TO DIGITAL DESIGN", Prentice-Hall Inc., 1980.

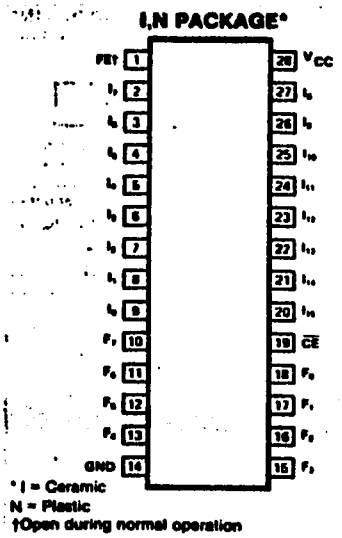
APPENDIX A

In this appendix,technical information about the Signetics 82S100 PLA is presented.

The data is extracted from ref. 12.

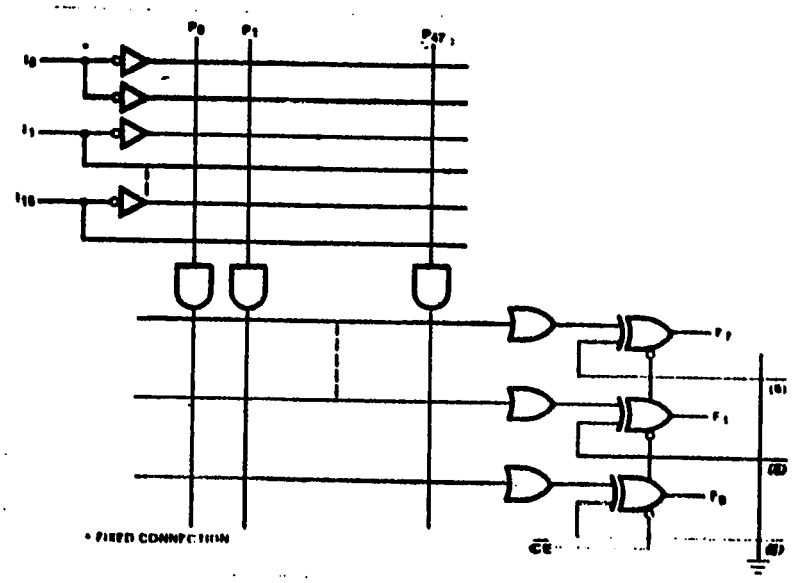
TABLE XXIII Electrical Characteristics

PARAMETER		MIN	TYP	MAX	UNIT
V_{CC}	Supply Voltage			+7	Vdc
I_{CC}	Supply Current		120	170	mA
T_A	Operating Temperature	0		75	°C
V_{IH}	High Input Voltage	2			V
V_{IL}	Low Input Voltage			0.85	V
V_{OH}	High Output Voltage	2.4			V
V_{OL}	Low Output Voltage		0.35	0.45	V
I_{IH}	High Input Current		<1	25	µA
I_{IL}	Low Input Current		-10	-100	µA
T_{IA}	Access Time from Input to Output		35	50	ns



MODE	Pn	$\overline{CE}$	Sr ? f(Pn)	Fp	$F_{\overline{p}}$
Disabled (82S101)	X	1	X	1	1
Disabled (82S100)				Hi-Z	Hi-Z
Read	1	0	Yes	1	0
	0	0		0	1
	X	0	No	0	1

Figure A.1. Pin Configuration and Truth Table.



Typical Product Term:
$P_0 = I_0 \cdot I_1 \cdot \overline{I_2} \cdot I_3 \cdot \overline{I_{13}}$

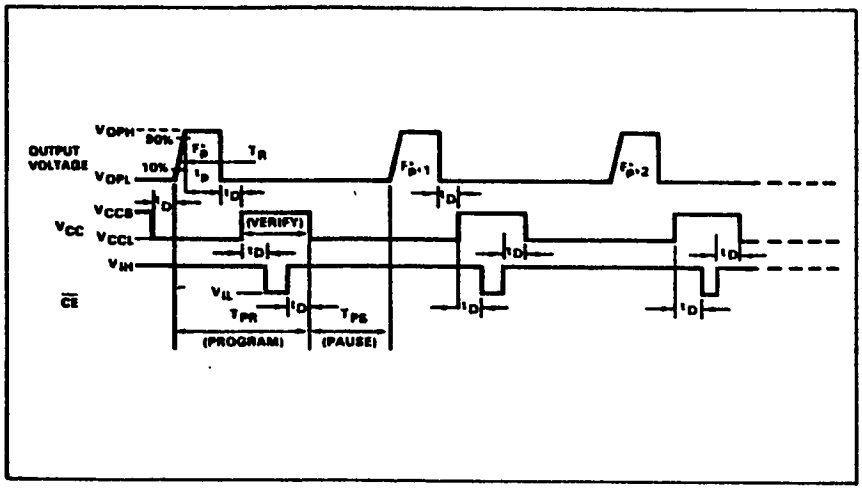
Typical Output Functions:
$F_0 = (\overline{CE}) + (P_0 + P_1 + P_2)$ @ S = Closed
$F_0 = (\overline{CE}) + (\overline{P_0} \cdot \overline{P_1} \cdot \overline{P_2})$ @ S = Open

NOTE

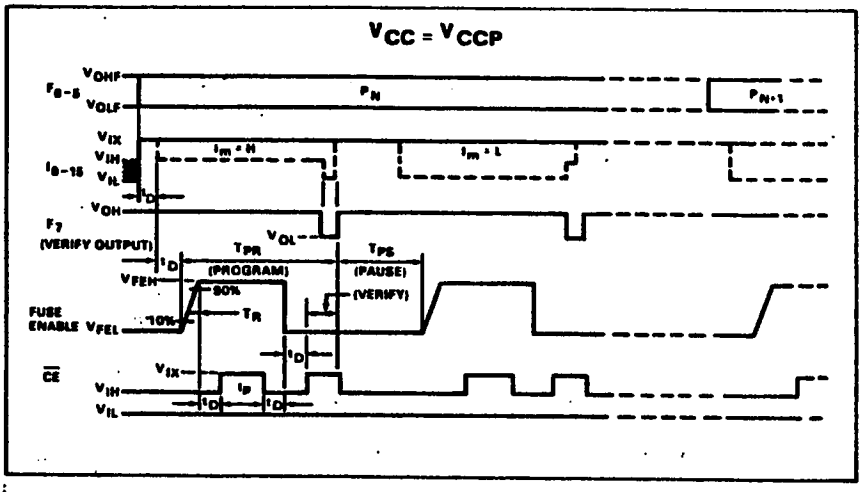
For each of the 8 outputs, either the function Fp (active-high) or $F\dot{p}$ (active low) is available, but not both. The required function polarity is programmed via link (fi).

Figure A.2. Logic Diagram and Logic Function.

OUTPUT POLARITY PROGRAM-VERIFY SEQUENCE (TYPICAL)



"AND" MATRIX PROGRAM-VERIFY SEQUENCE (TYPICAL)



"OR" MATRIX PROGRAM-VERIFY SEQUENCE (TYPICAL)

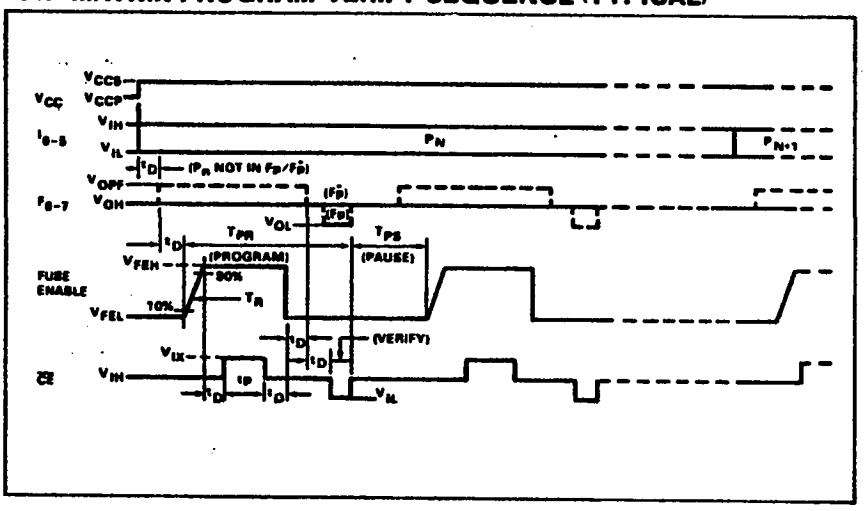


Figure A.3 Programming and Verifying Wave Forms .

TABLE XXIV Programming System Specifications•

PARAMETER		LIMITS Min	Typ	Max	UNIT
V_{CCS}	Vcc supply (program/verify "OR", verify output polarity)[2]	8.25	8.5	8.75	V
V_{CCI}	Vcc supply (program output polarity)	0	0.4	0.8	V
I_{CCS}	Icc limit (program "OR")	550		1,000	mA
	Output voltage				V
V_{OPH}	Program output polarity[3]	16.0	17.0	18.0	
V_{OPL}	Idle	0	0.4	0.8	
I_{OPH}	Output current limit (Program output polarity)	275	300	325	mA
	Input voltage				V
V_{IH}	High	2.4		5.5	
V_{IL}	Low	0	0.4	0.8	
	Input current				μA
I_{IH}	High			50	
I_{IL}	Low			-500	
	Forced output voltage				V
V_{OHF}	High	2.4		5.5	
V_{OLF}	Low	0	0.4	0.8	
	Output current				
I_{OHF}	High			100	μA
I_{OLF}	Low			-1	mA
V_{IX}	$\overline{CE}$ program enable level	9.5	10	10.5	V
I_{IX1}	Input variables current			2.5	mA
I_{IX2}	$\overline{CE}$ input current			5.0	mA
V_{FEH}	FE supply (program)[3]	16.0	17.0	18.0	V
V_{FEL}	FE supply (idle)	1.25	1.5	1.75	V
I_{FEH}	FE supply current limit	275	300	325	mA
V_{CCP}	Vcc supply (program/verify "AND")	4.75	5.0	5.25	V
I_{CCP}	Icc limit (program "AND")	550		1,000	mA
V_{OPF}	Forced output (program)	9.5	10	10.5	V
I_{OPF}	Output current (program)			10	mA
T_R	Output pulse rise time	10		50	μs
t_P	$\overline{CE}$ programming pulse width	0.3	0.4	0.5	ms
t_D	Pulse sequence delay	10			μs
T_{PR}	Programming time		0.6		ms
$\frac{T_{PR}}{T_{PR} + T_{PS}}$	Programming duty cycle			50	%
F_L	Fusing attempts per link			2	cycle
V_S	Verify threshold[4]	1.4	1.5	1.6	V

TABLE XXV Programming Steps for Various Procedures

PROCEDURE	FE	V_{CC}	$\overline{CE}$	I_0-I_{15}	F_0-F_7	PULSE WIDTH
OUTPUT	1 V_{FEL}					
	2	V_{CCL}				
	3		V_{IH}	V_{IH}		
	4				*(app) V_{OPH}	T_R+t_p
PRODUCT	1 V_{FEL}	V_{CCP}				
	2		V_{IH}			
	3			V_{IX}		
	4				**(add) V_{OHF}^{F0-F5}; V_{OLF}	
	5			(app) V_{IH};V_{IL}		
	6 V_{FEH}					
	7		V_{IX}			t_p
SUM	1 V_{FEL}					
	2		V_{IH}			
	3	V_{CCS}		V_{IX}		
	4			(add) V_{IH}^{I0-I5};V_{IL}		
	5				(app) V_{OPF}	
	6 V_{FEH}					
	7		V_{IX}			t_p
VERIFY OUTPUT	1 V_{FEL}	V_{CCS}				
	2		V_{IL}			
	3			V_{IH}		
	4				(app)sense	
VERIFY PRODUCT	1 V_{FEL}	V_{CCP}				
	2		V_{IX}			
	3			V_{IX}		
	4				(add) V_{OHF}^{F0-F5}; V_{OLF}	
	5			(app)V_{IH}+V_{IL}		
	6				sense F_7	
VERIFY SUM	1 V_{FEL}					
	2		V_{IH}			
	3	V_{CCS}		V_{IX}		
	4			(add) V_{IH}^{I0-I5};V_{IL}		
	5		V_{IL}			
	6				(app)sense	

* Selected Variables ** Address

Note: Time Difference Between a step and its preceding step is t_D.

APPENDIX B

USERS' MANUAL

The microcomputer control system is used to program the PLA's. The system is capable of programming and verifying the Signetics 82S 100/101 PLA's. In order to start the system the following initial steps should be performed (refer to Fig.B.1 for interconnections):

1. Switch off the CLR switch on the hardware board. The function of this switch is to reset all the microcomputer interface output signals.

2. Switch on the following hardware power supplies:
   a) The +17.5V @ 0.5A, +10.5V @ 0.5A, and +5.2V @ 2.5A.
   b) The 8.6V @ 1.5A power supply.

3. Switch on the following microcomputer power supplies:
   a) The -12V @ 510 mA,
   b) The +5V @ 2800 mA.

(Note: All voltages should be checked prior to system interconnection.)

4. Measure and readjust all voltages as needed.

5. Press the initialization key on the microcomputer board.

6. Put the CLR switch to ON.

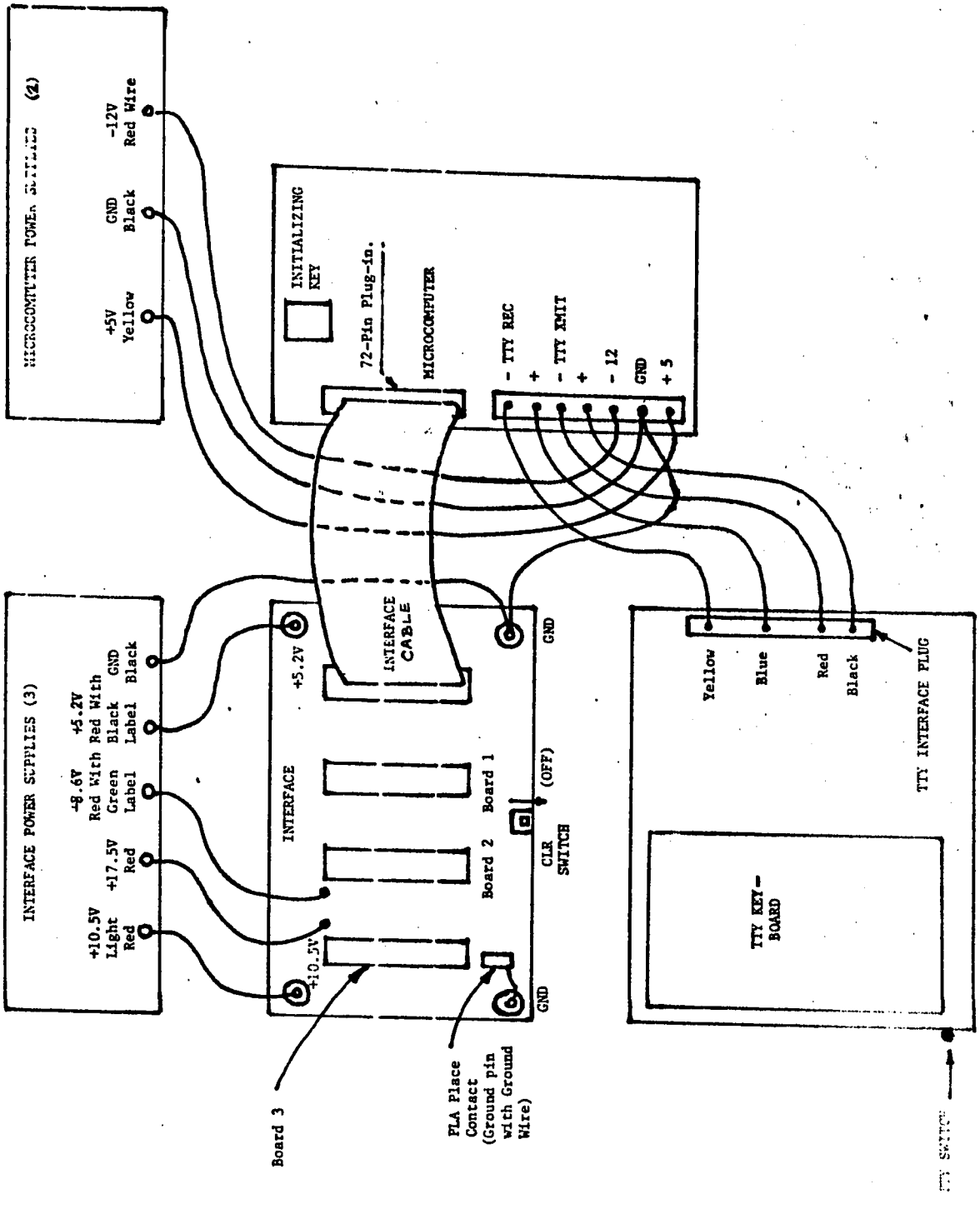


Figure B.1 System Connections.

When the system is to be switched off, the above steps are followed in the reverse sequence.

The use of the system to program/verify the PLA will be illustrated through examples.

B.1 From a Truth Table Specification

Assume that it is desired to program the following functions:

$f_0 = 1,3,5,9$

$\overline{f}_1 = 2,5,7$

The corresponding microcomputer input data is shown as in Fig. B.2. The first two characters of each row (AM) are used to load data into the microcomputer memory. The character is followed by the location at which the first data item in a row is to be loaded. The first row will have the following data:

1. An enumeration of the output function used. The number is computed in hexadecimal from the following bit organization:

← Not used →								f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0
X	X	X	X	X	X	X	X	0	0	0	0	0	0	1	1

AMØ4,3,5,6 ↵

AM4Ø,Ø1,FFFØ,1,1,3,FFFØ,1,2,5,FFFØ,3 ↵

AM4C,3,9,FFFØ,1,4,2,FFFØ,2,5,7,FFFØ,2 ↵

Note: ↵ means press the RETURN key,

Ø is the number zero.

Figure B.2 Input Data to the Microcomputer.

2. The output fun tions which are to be inverted and an inversion indicator (M) in the LSB position:

							f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0	M
X	X	X	X	X	X	X	0	0	0	0	0	0	1	0	1

3. The number of unique AND gates in hexadecimal (6_{16} for the example). Apart from the first row, the data provided defines which AND gates of a PLA to be used. This information is to be loaded starting at location 40_{16} of the microcomputer memory. Note in the example that only 12_{10} (C_{16}) hexadecimal words are loaded per line, the number after AM will increase by 12_{10} (C_{16}) for each line.

Each of the succeeding rows have information about the AND gates in the following general form:

AMX,a_J ,b_J ,c_J ,d_J ,a_{J+1} ,b_{J+1} ,c_{J+1} ,d_{J+1} ,a_{J+2} .....

Where:

X is the location in memory of the first data item in the row,

a_{J+i} is the AND gate number,

b_{J+i} indicates the AND gates literals (when a zero appears in a bit it means that associated variable is a complemented literal, otherwise it is uncomplemented.),

c_{J+i} identifies the ABD gate "don't care" variables (when the bit associated with a given variable is high, the variable appears as a "don't care" in the

AND gate.),

d_{j+i} designates which output functions are to be connected to the AND gate (when a one appears in a bit, this indicates that respective output function is connected to the AND gate.)

J is any uniquely void number between 0 and 47.

In the example provided, the AND gate labelled zero (a = 0) has input variable I_0 as an uncomplemented literal, variables I_1 through I_3 as complemented literals, and variables I_4 through I_{15} as "don't care cores" (c = FFFØ). Function zero is the only one containing AND gate zero ($d_J = 1$).

After providing the input data, the user may choose anyone of the characters shown in Table XXVI and type it.

If "BO" is chosen the output functions requiring inversion will be inverted and the burn-in of that function will be verified. If an error similar to that in Fig. B.3 a is typed by the system, it means that specified output was not inverted. In the provided error message, output 0 was supposed to be burned but it did not.

If "VO" is chosen, all the output functions' fuses will be verified that they are still intact. If an error message like that in Fig. B.3a is typed, it means that the output link specified is burned. Hence, the output F_0 link is burned as specified by the error message (Fig. B.3a).

When "BP" is chosen, the given AND gates, in the product

TABLE XXVI Input Symbols and Their Codes.

CODE	SYMBOL	DESCRIPTION
4241	BA	Execute all burn-in procedures and verify what has been burned using given data
5641	VA	Verify that the device is virgin
424F	BO	Execute the output burn-in and verify that output fuses indicated in the data are burned.
564F	VO	Verify that output fuses are still intact.
4250	BP	Execute the product burn-in and verify that product fuses specified in the data are burned.
5650	VP	Verify that product matrix fuses are intact.
4253	BS	Execute the sum burn-in and verify that burn-in is successful according to input data.
5653	VS	Verify that sum fuses are virgin.

ERROR *** OUTPUT Ø,

a. Output Error Message

ERROR *** PRD P-Ø2 IOI,1C,4C,6D,

b. Product Error Message

ERROR *** SUM P-Ø3 Ø,1,3,

c. Sum error message

Note: Ø is the number zero.

Figure B.3 System Error Messages.

matrix will be programmed and then verified. If an error message, like that in Fig. B.3b, is typed by the system, it means that the specified variables in the corresponding P-terms are not burned in accordance with supplied data. The error messages have the following general form:

(PRODUCT) (P-terms number x) (INPUTS {(I_K)(Condition of I_K); for $0 \leq K \leq 15$}

Where the input conditions are:

Uncomplemented	I
Complemented	C
Don't care	D
Both fuses are present	B

The error message shown in Fig. B.3b indicates that, while verifying product term 2, variable I_0 is uncomplemented, I_6 is a "don't care" and both I_1 and I_4 are in the complemented form.

When choosing "VP" the product matrix is verified to be virgin. If some element fuses are open, the system will type an error message like that in Fig. B.3b indicating the condition of those elements and their corresponding P-terms. The error message in Fig.B.3b indicates that variable I_0 is uncomplemented, I_1 and I_2 are complemented and I_6 is a "don't care". For all the remaining variables, fuses still exist.

Typing "BS" will burn specified fuses in the sum matrix and verify that those fuses are burned correctly. If some of the

these fuses are not bruned for the P-term, then the ouput for the unburned fuse will be typed as shown in Figure B.3c. The error message in Figure B.3c indicates that while verifying P-term 3, outputs F_0, F_1, and F_3 have fuses that were not burned as desired by the user.

Choosing the character "VS" will verify that the sum matrix is virgin. If it is not, the burned fuses and their respective P-terms will be printed like those in Figure B.3c. The error message (Figure B.3c) indicates that the fuses of outputs F_0, F_1 and F_3 are burned in the product term 3.

If the entire function is to be programmed by means of a single single user initialization, the character "BA" should be used. Character "VA" is used to verify that the entire device is virgin. The use of "VA" combines the separate computer operations for the characters "VO", "VP" and "VS". Similarly, characters "BO", "BP" and "BS" are combined in "BA". The verification characters ("V" prefix) can be used to give a list of the burned fuses in the device.

B.2 From a Minimization Program Output

If the minimization program is used, its output is in the format required by the microcomputer. Hence, no manual encoding is necessary and the data can be supplied directly to the microcomputer system. After that the user can choose any one character from Table XXVI and type it. Full details about the characters are provided in section B.1.

APPENDIX C

BURN-IN TESTS

Information relating to two burn-in tests for the system are contained within this appendix.

The first test is 1 of 4 detector. The minimization program was also used. The results are shown in Fig. C.1. The characters "VO", "VO", and "VS" of Fig.C.2 were used to verify that device is virgin. Characteris "BO", "BP" and "BS" are used to enact burn-in Characters "VO", "VP" and "VS" are used once more to list the fuses burned.

The second test was a decoder for a 7-segment display of hexadecimal Arabic/Latin characters. The minization program results are shown in Fig. C.3. In Fig. C.4, character "VA" was used to verify that device is virgin. Character "BA" was used to burn-in the fuses of the device. The second character "VA" was used to obtain a listing of the fuses burned in the device.

```
* * * * * * * * * * * * * * * * * * * * * * *

                    DATA

NO. OF VARIABLES= 4

NO. OF FUNCTIONS= 3

MAX. POSSIBLE NO. OF ANY- CASES=  1

INVERTED FUNCTIONS: 3
***FUNCTION- 1
NO. OF ALL MINTERMS=     1
NO. OF DONT-CARES=     0

                    MINTERMS:

          0
***FUNCTION- 2
NO. OF ALL MINTERMS=     4
NO. OF DONT-CARES=     0

                    MINTERMS:

          1     2     4     8
***FUNCTION- 3
NO. OF ALL MINTERMS=     5
NO. OF DONT-CARES=     0

                    MINTERMS:

          0     1     2     4     8
```

Figure C.1 Modified Minimization Program Output for Example 1.

```
* * * * * * * * * * * * * * * * * * * * *
                RESULTS:

NO. OF REQUIRED AND-GATES BEFORE MINIMIZATION=     5



TO FIND PI-S FOR PRODUCT OF 2-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 3,

          (    0 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F- 2, 3,

          (    1 ,     0)
          (    2 ,     0)
          (    4 ,     0)
          (    8 ,     0)

NO FURTHER MINIMIZATION COULD BE APPLIED

NO. OF AND-GATES=    5
          - - - - - - - - - - - - - - - - - - - -

AM04,0007,0009,0005

AM40,0000,0000,FFF0,0005,0001,0001,FFFC,0006,0002,0002,FFF0,0006

AM4C,0003,0004,FFF0,0006,0004,0008,FFFC,0006
```

Figure C.1 Modified Minimization Program Output for Example 1 (Cont'd).

```
>VO
>VP
>VS
>BO
>BP
>BS
>VO ERROR *** OUTPUT 2,
>VP ERROR *** PRD P-ØØ 1ØC,1C,2C,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø1 1Ø1,1C,2C,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø2 IØI,1I,2C,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø3 IØC,1C,2I,3C,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 ERROR *** PRD P-Ø4 IØC,1C,2C,3I,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
>VS ERROR *** SUM P-ØØ OUTPUT 1,
 ERROR *** SUM P-Ø1 OUTPUT Ø,
 ERROR *** SUM P-Ø2 OUTPUT Ø,
 ERROR *** SUM P-Ø3 OUTPUT Ø,
 ERROR *** SUM P-Ø4 OUTPUT Ø,
```

Figure C.2 I/O Data for Example 1 as Represented by System Software.

```
* * * * * * * * * * * * * * * * * * * * * *

                    DATA

NO. OF VARIABLES= 5

NO. OF FUNCTIONS= 7

MAX. POSSIBLE NO. OF AND- GATES=  1

***FUNCTION- 1
NO. OF ALL MINTERMS=   21
NO. OF DONT-CARES=    0

                    MINTERMS:

          0    2    3    5    6    7    8    9   10   12
         14   15   18   20   21   22   24   25   28   30
         31
***FUNCTION- 2
NO. OF ALL MINTERMS=   23
NO. OF DONT-CARES=    0

                    MINTERMS:

          0    4    5    6    8    9   10   11   12   14
         15   18   19   20   21   23   24   25   26   27
         28   30   31
***FUNCTION- 3
NO. OF ALL MINTERMS=   14
NO. OF DONT-CARES=    0

                    MINTERMS:

          2    3    4    5    6    8    9   10   11   13
         14   15   19   20   25   27   28   30   31
***FUNCTION- 4
NO. OF ALL MINTERMS=   21
NO. OF DONT-CARES=    0

                    MINTERMS:

          0    1    2    3    4    7    8    9   10   13
         17   19   21   22   23   24   25   27   28   30
         31
***FUNCTION- 5
NO. OF ALL MINTERMS=   19
NO. OF DONT-CARES=    0

                    MINTERMS:

          0    2    6    8   10   11   12   13   14   15
         18   19   20   21   23   24   26   28   30
***FUNCTION- 6
NO. OF ALL MINTERMS=   20
```

Figure C.3 Modified Minimization Program Output for Example 2.

```
NO. OF DONT-CARES=     0

                    MINTERMS:

          0    2    3    5    6    8    9   11   12   13
         14   16   20   21   23   27   28   29   30   31

***FUNCTION= 7
NO. OF ALL MINTERMS=    21
NO. OF DONT-CARES=      0

                    MINTERMS:

          0    1    3    4    5    6    7    8    9   10
         11   13   17   21   22   23   24   25   29   30
         31

* * * * * * * * * * * * * * * * * * * * *
                    RESULTS:

NO. OF REQUIRED AND-GATES BEFORE MINIMIZATION=   32
* * * * * PRIME IMPLICANTS COVERING FUNCTION= 6

          (   16 ,     4)

TO FIND PI-S FOR PRODUCT OF 2-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F= 2, 5,

          (    8 ,    22)
* * * * * PI-S COVERING PROD. FUNS. : F= 4, 7,

          (    1 ,    24)
* * * * * PI-S COVERING PROD. FUNS. : F= 6, 7,

          (   21 ,    10)
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TO FIND PI-S FOR PRODUCT OF 3-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 5,

          (   18 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 4, 7,

          (    3 ,     4)
          (   22 ,     8)



TO FIND PI-S FOR PRODUCT OF 4-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 5,

          (   14 ,     1)
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 5, 6,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 2, 3, 4, 5,

          (   19 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F- 2, 3, 4, 6,

          (   27 ,     4)
* * * * * PI-S COVERING PROD. FUNS. : F- 2, 3, 4, 7,

          (    4 ,     0)
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TO FIND PI-S FOR PRODUCT OF S-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F= 1, 2, 3, 4,
                                          7,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***

* * * * * PI-S COVERING PROD. FUNS. : F= 1, 2, 3, 5,
                                          6,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***

* * * * * PI-S COVERING PROD. FUNS. : F= 1, 2, 3, 6,
                                          7,
          (    5 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F= 1, 2, 4, 5,
                                          7,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***

* * * * * PI-S COVERING PROD. FUNS. : F= 1, 3, 4, 5,
                                          6,
          (    2 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F= 1, 3, 4, 6,
                                          7,
* * * * * PI-S COVERING PROD. FUNS. : F= 2, 3, 5, 6,
                                          7,
          (   11 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F= 2, 4, 5, 6,
                                          7,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***

* * * * * PI-S COVERING PROD. FUNS. : F= 3, 4, 5, 6,
                                          7,
          (   13 ,     0)
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TO FIND PI-S FOR PRODUCT OF 6-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 4,
                                          0,
*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 4,
                                          0,
*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 4,
                                          0,
*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 5,
                                          0,
          (    6 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 4, 5,
                                          0,
          (    0 ,     8)



TO FIND PI-S FOR PRODUCT OF 7-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 4,
                                          0, 7,
*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***

          * * * * * * * * * * * * * * * * * * * * * *
THE PROCESS IS REPEATED ONCE MORE
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 1

          (   31 ,     0)
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 3

          (    8 ,     2)
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 6

          (   12 ,    18)
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TO FIND PI-S FOR PRODUCT OF 2-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 6,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 3, 6,
          (    3 ,     0)



TO FIND PI-S FOR PRODUCT OF 3-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3,
          (    9 ,    16)
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 4, 7,
          (   24 ,     0)
* * * * * PI-S COVERING PROD. FUNS. : F- 2, 4, 5,
          (   21 ,     2)
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd)

```
TO FIND PI-S FOR PRODUCT OF 4-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 5,

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 3, 6,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 2, 4, 5,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 3, 4, 6,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 3, 4, 7,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***



TO FIND PI-S FOR PRODUCT OF 5-FUNCTIONS:



TO FIND PI-S FOR PRODUCT OF 6-FUNCTIONS:
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TC FIND PI-S FCR PRCDUCT OF 7-FUNCTIONS:

          * * * * * * * * * * * * * * * * * * * * *
THE PROCESS IS REPEATED ONCE MORE
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 1

          (   21 ,     0)
          (   12 ,    16)
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 3

          (   28 ,     2)
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 6

          (    9 ,     0)



TC FIND PI-S FCR PRODUCT OF 2-FUNCTIONS:
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TC FIND PI-S FOR PRODUCT OF 3-FUNCTIONS:

* * * * * PI-S COVERING PROD. FUNS. : F- 1, 3, 4,

*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP***
* * * * * PI-S COVERING PROD. FUNS. : F- 1, 4, 7,
          (   10 ,     0)


TC FIND PI-S FOR PRODUCT OF 4-FUNCTIONS:


TC FIND PI-S FOR PRODUCT OF 5-FUNCTIONS:


TO FIND PI-S FOR PRODUCT OF 6-FUNCTIONS:
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd).

```
TO FIND PI-S FOR PRODUCT OF 7-FUNCTIONS:

          * * * * * * * * * * * * * * * * * * * * * *
THE PROCESS IS REPEATED ONCE MORE
* * * * * PRIME IMPLICANTS COVERING FUNCTION- 4

          (   28 ,     01)

NO FURTHER MINIMIZATION COULD BE APPLIED

NO. OF AND-GATES=  31
          - - - - - - - - - - - - - - - - - - - - -

AM04,007F,0000,001F

AM40,0000,001C,FFE4,0020,0001,0008,FFF6,0012,0002,0001,FFF8,0048

AM4C,0003,0015,FFEA,0060,0004,0012,FFE0,0013,0005,0003,FFE4,0049

AM5E,0006,0016,FFE8,0049,0007,000E,FFE1,0017,0008,0013,FFE0,001E

AM64,0009,001B,FFE4,002E,000A,0004,FFE0,004E,000B,0005,FFE0,0067

AM7C,000C,0002,FFE0,003D,000D,000B,FFE0,0076,000E,000D,FFE0,007C

AM7C,000F,0006,FFE0,0077,0010,0000,FFE8,007B,0011,001F,FFE0,0001

AM86,0012,0006,FFE2,0004,0013,000C,FFF2,0020,0014,0003,FFE0,0024

AM94,0015,0009,FFF0,0007,0016,0018,FFE0,0049,0017,0015,FFE2,001A

AMA0,0018,0014,FFE0,0017,0019,0015,FFE0,0001,001A,000C,FFF0,0001

AMAC,001B,001C,FFE2,0004,001C,0009,FFE0,0020,001D,000A,FFE0,0049

AM86,001E,001C,FFE0,0008
```

Figure C.3 Modified Minimization Program Output for Example 2 (Cont'd)

```
>VA
>HA
>VA EHHOH *** EHD F-00 I0C,1C,2D,3C,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-01 I0C,1D,2D,3I,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-02 I0I,1C,2C,3D,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-03 I0I,1D,2I,3D,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-04 I0C,1I,2C,3C,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-05 I0I,1I,2D,3C,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-06 I0C,1I,2I,3D,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-07 I0D,1I,2I,3I,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-08 I0I,1I,2C,3C,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-09 I0I,1I,2D,3I,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-0A I0C,1C,2I,3C,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-0B I0I,1C,2I,3C,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-0C I0C,1I,2C,3C,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-0D I0I,1I,2C,3I,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-0E I0I,1C,2I,3I,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-0F I0C,1I,2I,3C,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-10 I0C,1C,2C,3D,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-11 I0I,1I,2I,3I,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-12 I0C,1D,2C,3I,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-13 I0C,1D,2I,3I,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-14 I0I,1I,2C,3C,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-15 I0I,1C,2C,3I,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-16 I0C,1C,2C,3I,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-17 I0I,1D,2I,3C,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-18 I0C,1C,2I,3C,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-19 I0I,1C,2I,3C,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-1A I0C,1C,2I,3I,4D,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-1B I0C,1D,2I,3I,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-1C I0I,1C,2C,3I,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-1D I0C,1I,2C,3I,4C,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** EHD F-1E I0C,1C,2I,3I,4I,5D,6D,7D,8D,9D,AD,BD,CD,DD,ED,FD,
 EHHOH *** SUM F-00 OUTPUT 0,1,2,3,4,6,
 EHHOH *** SUM F-01 OUTPUT 0,2,3,5,6,
 EHHOH *** SUM F-02 OUTPUT 0,1,2,4,5,
 EHHOH *** SUM F-03 OUTPUT 0,1,2,3,4,
 EHHOH *** SUM F-04 OUTPUT 2,3,5,6,
 EHHOH *** SUM F-05 OUTPUT 1,2,4,5,
 EHHOH *** SUM F-06 OUTPUT 1,2,4,5,
 EHHOH *** SUM F-07 OUTPUT 3,5,6,
```

Figure C.4 Output Data as Presented by the System Software.

```
EHHOH  *** SUM F-08 OUTPUT 0, 5, 6,
EHHOH  *** SUM F-09 OUTPUT 0, 4, 6,
EHHOH  *** SUM F-0A OUTPUT 0, 4, 5,
EHHOH  *** SUM F-0B OUTPUT 3, 4,
EHHOH  *** SUM F-0C OUTPUT 1, 6,
EHHOH  *** SUM F-0D OUTPUT 0, 3,
EHHOH  *** SUM F-0E OUTPUT 0, 1,
EHHOH  *** SUM F-0F OUTPUT 3,
EHHOH  *** SUM F-10 OUTPUT 2,
EHHOH  *** SUM F-11 OUTPUT 1, 2, 3, 4, 5, 6,
EHHOH  *** SUM F-12 OUTPUT 0, 1, 3, 4, 5, 6,
EHHOH  *** SUM F-13 OUTPUT 0, 1, 2, 3, 4, 6,
EHHOH  *** SUM F-14 OUTPUT 0, 1, 3, 4, 6,
EHHOH  *** SUM F-15 OUTPUT 3, 4, 5, 6,
EHHOH  *** SUM F-16 OUTPUT 1, 2, 4, 5,
EHHOH  *** SUM F-17 OUTPUT 0, 2, 5, 6,
EHHOH  *** SUM F-18 OUTPUT 3, 5, 6,
EHHOH  *** SUM F-19 OUTPUT 1, 2, 3, 4, 5, 6,
EHHOH  *** SUM F-1A OUTPUT 1, 2, 3, 4, 5, 6,
EHHOH  *** SUM F-1B OUTPUT 0, 1, 3, 4, 5, 6,
EHHOH  *** SUM F-1C OUTPUT 0, 1, 2, 3, 4, 6,
EHHOH  *** SUM F-1D OUTPUT 1, 2, 4, 5,
EHHOH  *** SUM F-1E OUTPUT 0, 1, 2, 4, 5, 6,

>
```

Figure C.4 Output Data as Presented by the System Software (Cont'd).

APPENDIX D

The following pages represent a listing of the microcomputer program. It includes the machine language listing and instruction locations in memory.

```
F400 5555 A          .WORD   05555          ;                    0001
F401 CD07 A          LD      AC3,TAD        ;                    0002
F402 F700 A LOOP:    SKNE    AC1,(AC3)      ;                    0003
F403 9B01 A          JMP     @01(AC3)       ;                    0004
F404 C300 A          LD      AC0,(AC3)      ;                    0005
F405 4102 A          BEC     AC0<>0,ERR     ;                    0006
F406 7B02 A          AISZ    AC3,+2         ;                    0007
F407 19FA A          JMP     LOOP           ;                    0008
F408 8000 A ERR:     RTS                    ;                    0009
F409 F40A A TAD:     .WORD   0F40A          ;                    0010
F40A 4241 A          .WORD   04241          ;                    0011
F40B F41B A          .WORD   0F41B          ;                    0012
F40C 5641 A          .WORD   05641          ;                    0013
F40D F424 A          .WORD   0F424          ;                    0014
F40E 424F A          .WORD   0424F          ;                    0015
F40F F42D A          .WORD   0F42D          ;                    0016
F410 564F A          .WORD   0564F          ;                    0017
F411 F430 A          .WORD   0F430          ;                    0018
F412 4250 A          .WORD   04250          ;                    0019
F413 F433 A          .WORD   0F433          ;                    0020
F414 5650 A          .WORD   05650          ;                    0021
F415 F436 A          .WORD   0F436          ;                    0022
F416 4253 A          .WORD   04253          ;                    0023
F417 F439 A          .WORD   0F439          ;                    0024
F418 5653 A          .WORD   05653          ;                    0025
F419 F43C A          .WORD   0F43C          ;                    0026
F41A 0000 A          .WORD   00000          ;                    0027
F41B 1523 A BA:      JSR     INT            ;                    0028
F41C 952E A          JSR     @OUT           ;                    0029
F41D 5C00 A          NOP                    ;                    0030
F41E 5C00 A          NOP                    ;                    0031
F41F 952D A          JSR     @PROD          ;                    0032
F420 5C00 A          NOP                    ;                    0033
F421 5C00 A          NOP                    ;                    0034
F422 952C A          JSR     @SUM           ;                    0035
F423 1922 A          JMP     RET            ;                    0036
F424 151A A VA:      JSR     INT            ;                    0037
F425 9526 A          JSR     @OUTV          ;                    0038
F426 5C00 A          NOP                    ;                    0039
F427 5C00 A          NOP                    ;                    0040
F428 9525 A          JSR     @PRODV         ;                    0041
F429 5C00 A          NOP                    ;                    0042
F42A 5C00 A          NOP                    ;                    0043
F42B 9524 A          JSR     @SUMV          ;                    0044
F42C 1919 A          JMP     RET            ;                    0045
F42D 1511 A BO:      JSR     INT            ;                    0046
F42E 951C A          JSR     @OUT           ;                    0047
F42F 1916 A          JMP     RET            ;                    0048
F430 150E A VO:      JSR     INT            ;                    0049
F431 951A A          JSR     @OUTV          ;                    0050
F432 1913 A          JMP     RET            ;                    0051
F433 150B A BP:      JSR     INT            ;                    0052
F434 9518 A          JSR     @PROD          ;                    0053
F435 1910 A          JMP     RET            ;                    0054
F436 1508 A VP:      JSR     INT            ;                    0055
F437 9516 A          JSR     @PRODV         ;                    0056
F438 190D A          JMP     RET            ;                    0057
F439 1505 A BS:      JSR     INT            ;                    0058
F43A 9514 A          JSR     @SUM           ;                    0059
```

```
F43B 190A A          JMP     RET             ;                       0060
F43C 1502 A VS:      JSR     INT             ;                       0061
F43D 9512 A          JSR     @SUMV           ;                       0062
F43E 1907 A          JMP     RET             ;                       0063
F43F D803 A INT:     ST      AC2,3           ;                       0064
F440 5C00 A          NOP                     ;                       0065
F441 5C00 A          NOP                     ;                       0066
F442 5000 A          LD      AC0,0                                   0067
F443 B10D A          ST      AC0,ADA         ;                       0068
F444 C905 A          LD      AC2,AD4         ;                       0069
F445 8000 A          RTS                     ;                       0070
F446 9502 A RET:     JSR     @CRLF           ;                       0071
F447 C803 A          LD      AC2,3           ;                       0072
F448 8001 A          RTS     +1              ;                       0073
F449 F26B A CRLF:    .WORD   0F26B           ;                       0074
F44A 0104 A AD4:     .WORD   00104           ;                       0075
F44B F460 A OUT:     .WORD   0F460           ;                       0076
F44C F4AE A OUTV:    .WORD:  0F4AE           ;                       0077
F44D F4CB A PROD:    .WORD   0F4CB           ;                       0078
F44E F56E A PRODV:   .WORD   0F59E           ;                       0079
F44F F59C A SUM:     .WORD   0F59C           ;                       0080
F450 F60B A SUMV:    .WORD   0F60B           ;                       0081
F451 010A A ADA:     .WORD   0010A           ;                       0082
F452 0000 A                                  ;                       0083
F453 0000 A                                  ;                       0084
F454 0000 A                                  ;                       0085
F455 0000 A                                  ;                       0086
F456 0000 A                                  ;                       0087
F457 0000 A                                  ;                       0088
F458 0000 A                                  ;                       0089
F459 0000 A                                  ;                       0090
F45A 0000 A                                  ;                       0091
F45B 0000 A                                  ;                       0092
F45C 0000 A                                  ;                       0093
F45D 0000 A                                  ;                       0094
F45E 0000 A                                  ;                       0095
F45F 0000 A                                  ;                       0096
F460 C005 A OUT:     LD      AC0,5           ;                       0097
F161 4301 A          BCC     BIT0,+2         ;                       0098
F162 8000 A          RTS                     ;                       0099
F463 CD2D A          LD      AC3,AD1         ;                       0100
F464 D301 A          ST      AC0,001(AC3)    ;                       0101
F465 5000 A          LI      AC0,0           ;                       0102
F466 D300 A          ST      AC0,(AC3)       ;                       0103
F467 C12C A          LD      AC0,D1          ;                       0104
F468 B129 A          ST      AC0,@AD2        ;                       0105
F469 C128 A          LD      AC0,D2          ;                       0106
F46A B127 A          ST      AC0,@AD2        ;                       0107
F46B C12A A          LD      AC0,D3          ;                       0108
F46C B125 A          ST      AC0,@AD2        ;                       0109
F46D 9530 A LOOP:    JSR     @SHIFT1         ;                       0110
F46E 9530 A          JSR     @CHECK          ;                       0111
F46F 1919 A          JMP     AGAIN           ;                       0112
F470 C300 A          LD      AC0,(AC3)       ;                       0113
F471 B121 A          ST      AC0,@AD3        ;                       0114
F472 C124 A          LD      AC0,D4          ;                       0115
F473 B11E A          ST      AC0,@AD2        ;                       0116
F474 5111 A          LI      AC1,11          ;                       0117
F475 952A A          JSR     @DELAY          ;                       0118
```

```
F476 11F  A           LD      AC0,D3          ;                       0119
F477 B11A A           ST      AC0,@AD2        ;                       0120
F478 C11F A           LD      AC0,D5          ;                       0121
F479 B11A A           ST      AC0,@AD2        ;                       0122
F47A C11E A           LD      AC0,D6          ;                       0123
F47B B116 A           ST      AC0,@AD2        ;                       0124
F47C C11F A           LD      AC0,D9          ;                       0125
F47D B114 A           ST      AC0,@AD2        ;                       0126
F47E 1529 A           JSR     OVER            ;                       0127
F47F C119 A           LD      AC0,D6          ;                       0128
F480 B111 A           ST      AC0,@AD2        ;                       0129
F481 C116 A           LD      AC0,D5          ;                       0130
F482 B10F A           ST      AC0,@AD2        ;                       0131
F483 C112 A           LD      AC0,D3          ;                       0132
F484 B10D A           ST      AC0,@AD2        ;                       0133
F485 951C A           JSR     @SENCH          ;                       0134
F486 951C A           JSR     @ERST           ;                       0135
F487 5115 A           LI      AC1,15          ;                       0136
F488 9517 A           JSR     @DELAY          ;                       0137
F489 951A A AGAIN:    JSR     @INC1           ;                       0138
F48A 19E2 A           JMP     LOOP            ;                       0139
F48B 9519 A           JSR     @PRINT          ;                       0140
F48C 9519 A           JSR     @EOUT           ;                       0141
F48D C107 A           LD      AC0,D2          ;                       0142
F48E B103 A           ST      AC0,@AD2        ;                       0143
F48F C104 A           LD      AC0,D1          ;                       0144
F490 9516 A           JMP     @FIN            ;                       0145
F491 0100 A AD1:      .WORD   0C100           ;                       0146
F492 9000 A AD2:      .WORD   09000           ;                       0147
F493 A000 A AD3:      .WORD   0A000           ;                       0148
F494 0110 A D1:       .WORD   00110           ;                       0149
F495 4114 A D2:       .WORD   04114           ;                       0150
F496 4954 A D3:       .WORD   04954           ;                       0151
F497 49D4 A D4:       .WORD   049D4           ;                       0152
F498 4950 A D5:       .WORD   04950           ;                       0153
F499 4852 A D6:       .WORD   04852           ;                       0154
F49A 0012 A D7:       .WORD   00012           ;                       0155
F49B 0812 A D8:       .WORD   00812           ;                       0156
F49C 0852 A D9:       .WORD   00852           ;                       0157
F49D 0C52 A D10:      .WORD   00C52           ;                       0158
F49E F6AC A SHIFT1:   .WORD   0F6AC           ;                       0159
F49F F641 A CHECK:    .WORD   0F641           ;                       0160
F4A0 F2FF A DELAY:    .WORD   0F2FF           ;                       0161
F4A1 F6BA A SENSE1:   .WORD   0F6BA           ;                       0162
F4A2 F640 A SENCH:    .WORD   0F640           ;                       0163
F4A3 F694 A ERST:     .WORD   0F694           ;                       0164
F4A4 F63B A INC1:     .WORD   0F63B           ;                       0165
F4A5 F644 A PRINT:    .WORD   0F644           ;                       0166
F4A6 F655 A EOUT:     .WORD   0F655           ;                       0167
F4A7 F6C6 A FIN:      .WORD   0F6C6           ;                       0168
F4A8 C1F4 A OVER:     LD      AC0,D10         ;                       0169
F4A9 B1E8 A           ST      AC0,@AD2        ;                       0170
F4AA 95F6 A           JSR     @SENSE1         ;                       0171
F4AB C1F0 A           LD      AC0,D9          ;                       0172
F4AC B1E5 A           ST      AC0,@AD2        ;                       0173
F4AD 8000 A           RTS                     ;                       0174
F4AE 5000 A OUTV:     LI      AC0,0           ;                       0175
F4AF CDE1 A           LD      AC3,AD1         ;                       0176
F4B0 D300 A           ST      AC0,(AC3)       ;                       0177
```

```
F4B1 D309 A          ST      AC0,004(AC3)    ;                       0178
F4B2 C1E7 A          LD      AC0,D7          ;                       0179
F4B3 B1DE A          ST      AC0,@AD2        ;                       0180
F4B4 C1E6 A          LD      AC0,D9          ;                       0181
F4B5 B1DC A          ST      AC0,@AD2        ;                       0182
F4B6 C1E5 A          LD      AC0,D9          ;                       0183
F4B7 B1DA A          ST      AC0,@AD2        ;                       0184
F4B8 C300 A LOOP:    LD      AC0,(AC3)       ;                       0185
F4B9 B1D9 A          ST      AC0,@AD3        ;                       0186
F4BA 15ED A          JSR     OVER            ;                       0187
F4BB 95E6 A          JSR     @SENCH          ;                       0188
F4BC 1901 A          JMP     +2              ;                       0189
F4BD 95E5 A          JSR     @ERST           ;                       0190
F4BE C305 A          LD      AC0,005(AC3)    ;                       0191
F4BF 2810 A          SHL     AC0,8,0         ;                       0192
F4C0 E3C9 A          ADD     AC0,009(AC3)    ;                       0193
F4C1 2C02 A          SHR     AC0,1,0         ;                       0194
F4C2 D309 A          ST      AC0,009(AC3)    ;                       0195
F4C3 95E0 A AGAIN:   JSR     @INC1           ;                       0196
F4C4 19F3 A          JMP     LOOP            ;                       0197
F4C5 95DF A          JSR     @PRINT          ;                       0198
F4C6 95DF A          JSR     @EOUT           ;                       0199
F4C7 C1D3 A          LD      AC0,D8          ;                       0200
F4C8 B1C9 A          ST      AC0,@AD2        ;                       0201
F4C9 C1D0 A          LD      AC0,D7          ;                       0202
F4CA 95DC A          JMP     @FIN            ;                       0203
F4CB C006 A PROD:    LD      AC0,6           ;                       0204
F4CC CDC4 A          LD      AC3,AD1         ;                       0205
F4CD 5240 A          LI      AC2,40          ;                       0206
F4CE D304 A          ST      AC0,004(AC3)    ;                       0207
F4CF C14D A          LD      AC0,D11         ;                       0208
F4D0 B1C1 A          ST      AC0,@AD2        ;                       0209
F4D1 C14C A          LD      AC0,D12         ;                       0210
F4D2 B1BF A          ST      AC0,@AD2        ;                       0211
F4D3 C14B A          LD      AC0,D13         ;                       0212
F4D4 B1BD A          ST      AC0,@AD2        ;                       0213
F4D5 C200 A LOOP1:   LD      AC0,(AC2)       ;                       0214
F4D6 2810 A          SHL     AC0,8,0         ;                       0215
F4D7 B1B6 A          ST      AC0,@AD3        ;                       0216
F4D8 A54E A          OR      AC0,S1          ;                       0217
F4D9 B1B9 A          ST      AC0,@AD3        ;                       0218
F4DA D307 A          ST      AC0,007(AC3)    ;                       0219
F4DB 5000 A          LI      AC0,0           ;                       0220
F4DC D300 A          ST      AC0,(AC3)       ;                       0221
F4DD C202 A          LD      AC0,002(AC2)    ;                       0222
F4DE D302 A          ST      AC0,002(AC3)    ;                       0223
F4DF C201 A          LD      AC0,001(AC2)    ;                       0224
F4E0 D301 A          ST      AC0,001(AC3)    ;                       0225
F4E1 C302 A LOOP2:   LD      AC0,002(AC3)    ;                       0226
F4E2 430D A          BOC     BIT0,BOTH       ;                       0227
F4E3 C301 A          LD      AC0,001(AC3)    ;                       0228
F4E4 4305 A          BOC     BIT0,I          ;                       0229
F4E5 5001 A IB:      LI      AC0,1           ;                       0230
F4E6 D308 A          ST      AC0,008(AC3)    ;                       0231
F4E7 C307 A          LD      AC0,007(AC3)    ;                       0232
F4E8 154F A          JSR     BURN            ;                       0233
F4E9 1910 A          JMP     V-3             ;                       0234
F4EA 5002 A I:       LI      AC0,2           ;                       0235
F4EB D308 A          ST      AC0,006(AC3)    ;                       0236
```

```
F4EC C307 A          LD      AC0,007(AC3)    ;                       0237
F4ED A53A A          OR      AC0,B1          ;                       0238
F4EE 1549 A          JSR     BURN            ;                       0239
F4EF 1900 A          JMP     V               ;                       0240
F4F0 5003 A BOTH:    LI      AC0,3           ;                       0241
F4F1 D308 A          ST      AC0,008(AC3)    ;                       0242
F4F2 C307 A          LD      AC0,007(AC3)    ;                       0243
F4F3 A534 A          OR      AC0,B1          ;                       0244
F4F4 1543 A          JSR     BURN            ;                       0245
F4F5 5115 A          LI      AC1,15          ;                       0246
F4F6 9540 A          JSR     @DELAY          ;                       0247
F4F7 C307 A          LD      AC0,007(AC3)    ;                       0248
F4F8 A931 A          AND     AC0,B2          ;                       0249
F4F9 1541 A          JSR     BURN+3          ;                       0250
F4FA C307 A          LD      AC0,007(AC3)    ;                       0251
F4FB A52C A          OR      AC0,B1          ;                       0252
F4FC 951D A          JSR     @STV            ;                       0253
F4FD C124 A V:       LD      AC0,D16         ;                       0254
F4FE B11C A          ST      AC0,@AD2        ;                       0255
F4FF 1548 A          JSR     PVER            ;                       0256
F500 C121 A          LD      AC0,D16         ;                       0257
F501 B119 A          ST      AC0,@AD2        ;                       0258
F502 C11C A          LD      AC0,D13         ;                       0259
F503 B117 A          ST      AC0,@AD2        ;                       0260
F504 154C A          JSR     RESET           ;                       0261
F505 5118 A          LI      AC1,18          ;                       0262
F506 9530 A          JSR     @DELAY          ;                       0263
F507 9527 A AGAIN:   JSR     @INC2           ;                       0264
F508 1901 A          JMP     +2              ;                       0265
F509 1902 A          JMP     DEC             ;                       0266
F50A 9523 A          JSR     @SHIFT2         ;                       0267
F50B 1905 A          JMP     LOOP2           ;                       0268
F50C 9524 A DEC:     JSR     @PRINT          ;                       0269
F50D 9528 A          JSR     @EPRD           ;                       0270
F50E C307 A          LD      AC0,007(AC3)    ;                       0271
F50F A91D A          AND     AC0,S4          ;                       0272
F510 B10B A          ST      AC0,@AD3        ;                       0273
F511 AF04 A          DSZ     004(AC3)        ;                       0274
F512 1901 A          JMP     +2              ;                       0275
F513 1902 A          JMP     +3              ;                       0276
F514 7A04 A          AISZ    AC2,4           ;                       0277
F515 19BF A          JMP     LOOP1           ;                       0278
F516 C107 A END:     LD      AC0,D12         ;                       0279
F517 B103 A          ST      AC0,@AD2        ;                       0280
F518 C104 A          LD      AC0,D11         ;                       0281
F519 991B A          JMP     @FIN            ;                       0282
F51A F6CB A STV:     .WORD   0F6CB           ;                       0283
F51B 9000 A AD2:     .WORD   09000           ;                       0284
F51C A000 A AD3:     .WORD   0A000           ;                       0285
F51D 4011 A D11:     .WORD   04011           ;                       0286
F51E 4811 A D12:     .WORD   04811           ;                       0287
F51F 5811 A D13:     .WORD   05811           ;                       0288
F520 5819 A D14:     .WORD   05819           ;                       0289
F521 5839 A D15:     .WORD   05839           ;                       0290
F522 5831 A D16:     .WORD   05831           ;                       0291
F523 7831 A D17:     .WORD   07831           ;                       0292
F524 0011 A D18:     .WORD   00011           ;                       0293
F525 0031 A D19:     .WORD   00031           ;                       0294
F526 1031 A D20:     .WORD   01031           ;                       0295
```

```
F527 0020 A S1:       .WORD   00020           ;                       0296
F528 4000 A B1:       .WORD   04000           ;                       0297
F529 0060 A S2:       .WORD   00060           ;                       0298
F52A 3FFF A B2:       .WORD   03FFF           ;                       0299
F52B 3F2F A S3:       .WORD   03F2F           ;                       0300
F52C 3F20 A B3:       .WORD   03F20           ;                       0301
F52D 3F00 A S4:       .WORD   03F00           ;                       0302
F52E F6A9 A SHIFT2:   .WORD   0F6A9           ;                       0303
F52F F639 A INC2:     .WORD   0F639           ;                       0304
F530 F6BE A SENSE2:   .WORD   0F6BE           ;                       0305
F531 F644 A PRINT:    .WORD   0F644           ;                       0306
F532 F694 A ERST:     .WORD   0F694           ;                       0307
F533 F6BA A SENSE1:   .WORD   0F6BA           ;                       0308
F534 F634 A INC3:     .WORD   0F634           ;                       0309
F535 F6C6 A FIN:      .WORD   0F6C6           ;                       0310
F536 F64F A EPRD:     .WORD   0F64F           ;                       0311
F537 F2FF A DELAY:    .WORD   0F2FF           ;                       0312
F538 A700 A BURN:     OR      AC0,(AC3)       ;                       0313
F539 B1E2 A           ST      AC0,@AD3        ;                       0314
F53A A5EE A           OR      AC0,S2          ;                       0315
F53B B1E0 A           ST      AC0,@AD3        ;                       0316
F53C D307 A           ST      AC0,007(AC3)    ;                       0317
F53D C1E2 A           LD      AC0,D14         ;                       0318
F53E B1DC A           ST      AC0,@AD2        ;                       0319
F53F C1E1 A           LD      AC0,D15         ;                       0320
F540 B1DA A           ST      AC0,@AD2        ;                       0321
F541 5111 A           LI      AC1,11          ;                       0322
F542 95F4 A           JSR     @DELAY          ;                       0323
F543 C1DC A           LD      AC0,D14         ;                       0324
F544 B1D6 A           ST      AC0,@AD2        ;                       0325
F545 C1D9 A           LD      AC0,D13         ;                       0326
F546 B1D4 A           ST      AC0,@AD2        ;                       0327
F547 8000 A           RTS                     ;                       0328
F548 C1DA A PVER:     LD      AC0,D17         ;                       0329
F549 B1D1 A           ST      AC0,@AD2        ;                       0330
F54A 95E5 A           JSR     @SENSE2         ;                       0331
F54B C307 A           LD      AC0,007(AC3)    ;                       0332
F54C A9DD A           AND     AC0,B2          ;                       0333
F54D B1CE A           ST      AC0,@AD3        ;                       0334
F54E D307 A           ST      AC0,007(AC3)    ;                       0335
F54F 95E3 A           JSR     @SENSE1         ;                       0336
F550 8000 A           RTS                     ;                       0337
F551 C307 A RESET:    LD      AC0,007(AC3)    ;                       0338
F552 A9D8 A           AND     AC0,S3          ;                       0339
F553 B1C8 A           ST      AC0,@AD3        ;                       0340
F554 A9D7 A           AND     AC0,B3          ;                       0341
F555 D307 A           ST      AC0,007(AC3)    ;                       0342
F556 C306 A           LD      AC0,006(AC3)    ;                       0343
F557 2802 A           SHL     AC0,1,0         ;                       0344
F558 E305 A           ADD     AC0,005(AC3)    ;                       0345
F559 D305 A           ST      AC0,005(AC3)    ;                       0346
F55A F308 A           SKNE    AC0,008(AC3)    ;                       0347
F55B 8000 A           RTS                     ;                       0348
F55C 95D5 A           JSR     @ERST           ;                       0349
F55D C305 A           LD      AC0,005(AC3)    ;                       0350
F55E 4305 A           BDC     BIT0,+6         ;                       0351
F55F 4402 A           BDC     BIT1,+3         ;                       0352
F560 5042 A           LI      AC0,42          ;                       0353
F561 1906 A           JMP     +7              ;                       0354
```

```
F562 5049 A          LI      AC0,49          ;                       0355
F563 1904 A          JMP     +5              ;                       0356
F564 4402 A          BOC     BIT1,+3         ;                       0357
F565 5043 A          LI      AC0,43          ;                       0358
F566 1901 A          JMP     +2              ;                       0359
F567 5044 A          LI      AC0,44          ;                       0360
F568 2808 A          SHL     AC0,4,0         ;                       0361
F569 6E40 A          RXCH    AC1,AC2         ;                       0362
F56A E200 A          ADD     AC0,(AC2)       ;                       0363
F56B D200 A          ST      AC0,(AC2)       ;                       0364
F56C 6D80 A          RXCH    AC2,AC1         ;                       0365
F56D 8000 A          RTS                     ;                       0366
F56E 5000 A PRODV:   LI      AC0,0           ;                       0367
F56F C071 A          LD      AC3,AD1         ;                       0368
F570 D304 A          ST      AC0,004(AC3)    ;                       0369
F571 D308 A          ST      AC0,008(AC3)    ;                       0370
F572 C1B1 A          LD      AC0,D18         ;                       0371
F573 B1A7 A          ST      AC0,@AD2        ;                       0372
F574 C1B0 A          LD      AC0,D19         ;                       0373
F575 B1A5 A          ST      AC0,@AD2        ;                       0374
F576 C1AF A          LD      AC0,D20         ;                       0375
F577 B1A3 A          ST      AC0,@AD2        ;                       0376
F578 C304 A LOOP1:   LD      AC0,004(AC3)    ;                       0377
F579 2810 A          SHL     AC0,8,0         ;                       0378
F57A B1A1 A          ST      AC0,@AD3        ;                       0379
F57B A5AB A          OR      AC0,S1          ;                       0380
F57C B19F A          ST      AC0,@AD3        ;                       0381
F57D D307 A          ST      AC0,007(AC3)    ;                       0382
F57E 5000 A          LI      AC0,0           ;                       0383
F57F D300 A          ST      AC0,(AC3)       ;                       0384
F580 C307 A LOOP2:   LD      AC0,007(AC3)    ;                       0385
F581 A700 A          OR      AC0,(AC3)       ;                       0386
F582 B199 A          ST      AC0,@AD3        ;                       0387
F583 A5A4 A          OR      AC0,B1          ;                       0388
F584 B197 A          ST      AC0,@AD3        ;                       0389
F585 A5A3 A          OR      AC0,S2          ;                       0390
F586 B195 A          ST      AC0,@AD3        ;                       0391
F587 D307 A          ST      AC0,007(AC3)    ;                       0392
F588 15BF A          JSR     PVER            ;                       0393
F589 C19C A          LD      AC0,D20         ;                       0394
F58A B190 A          ST      AC0,@AD2        ;                       0395
F58B 15C5 A          JSR     RESET           ;                       0396
F58C 95A2 A AGAIN:   JSR     @INC2           ;                       0397
F58D 19F2 A          JMP     LOOP2           ;                       0398
F58E 5C00 A          NOP                     ;                       0399
F58F 5C00 A          NOP                     ;                       0400
F590 5C00 A          NOP                     ;                       0401
F591 959F A DEC:     JSR     @PRINT          ;                       0402
F592 95A3 A          JSR     @EPRD           ;                       0403
F593 C307 A          LD      AC0,007(AC3)    ;                       0404
F594 A998 A          AND     AC0,S4          ;                       0405
F595 B186 A          ST      AC0,@AD3        ;                       0406
F596 959D A          JSR     @INC3           ;                       0407
F597 19E0 A          JMP     LOOP1           ;                       0408
F598 C18C A END:     LD      AC0,D19         ;                       0409
F599 B181 A          ST      AC0,@AD2        ;                       0410
F59A C189 A          LD      AC0,D18         ;                       0411
F59B 9999 A          JMP     @FIN            ;                       0412
F59C C006 A SUM:     LD      AC0,6           ;                       0413
```

```
F59D 5240 A          LI      AC2,40          ;                       0414
F59E CD42 A          LD      AC3,AD1         ;                       0415
F59F D304 A          ST      AC0,004(AC3)    ;                       0416
F5A0 155C A          JSR     SM1             ;                       0417
F5A1 C004 A LOOP1:   LD      AC0,4           ;                       0418
F5A2 D301 A          ST      AC0,001(AC3)    ;                       0419
F5A3 C005 A          LD      AC0,5           ;                       0420
F5A4 2C02 A          SHR     AC0,1,0         ;                       0421
F5A5 D302 A          ST      AC0,002(AC3)    ;                       0422
F5A6 C200 A          LD      AC0,(AC2)       ;                       0423
F5A7 2610 A          SHL     AC0,8,0         ;                       0424
F5A8 1551 A          JSR     SM2             ;                       0425
F5A9 C203 A          LD      AC0,003(AC2)    ;                       0426
F5AA D303 A          ST      AC0,003(AC3)    ;                       0427
F5AB C301 A LOOP2:   LD      AC0,001(AC3)    ;                       0428
F5AC 4301 A          BOC     BIT0,+2         ;                       0429
F5AD 191F A          JMP     AGAIN           ;                       0430
F5AE C303 A          LD      AC0,003(AC3)    ;                       0431
F5AF 431D A          BOC     BIT0,AGAIN      ;                       0432
F5B0 C307 A          LD      AC0,007(AC3)    ;                       0433
F5B1 A700 A          OR      AC0,(AC3)       ;                       0434
F5B2 B130 A          ST      AC0,@AD3        ;                       0435
F5B3 D307 A          ST      AC0,007(AC3)    ;                       0436
F5B4 C132 A          LD      AC0,D24         ;                       0437
F5B5 B12C A          ST      AC0,@AD2        ;                       0438
F5B6 C131 A          LD      AC0,D25         ;                       0439
F5B7 B12A A          ST      AC0,@AD2        ;                       0440
F5B8 C130 A          LD      AC0,D26         ;                       0441
F5B9 B128 A          ST      AC0,@AD2        ;                       0442
F5BA 5111 A          LI      AC1,11          ;                       0443
F5BB 9533 A          JSR     @DELAY          ;                       0444
F5BC C129 A          LD      AC0,D25         ;                       0445
F5BD B124 A          ST      AC0,@AD2        ;                       0446
F5BE C128 A          LD      AC0,D24         ;                       0447
F5BF B122 A          ST      AC0,@AD2        ;                       0448
F5C0 C125 A          LD      AC0,D23         ;                       0449
F5C1 B120 A          ST      AC0,@AD2        ;                       0450
F5C2 153E A          JSR     SVER            ;                       0451
F5C3 C302 A          LD      AC0,002(AC3)    ;                       0452
F5C4 4304 A          BOC     BIT0,+5         ;                       0453
F5C5 157A A          JSR     SENCH           ;                       0454
F5C6 1904 A          JMP     +5              ;                       0455
F5C7 952A A          JSR     @ERST           ;                       0456
F5C8 1902 A          JMP     +3              ;                       0457
F5C9 1576 A          JSR     SENCH           ;                       0458
F5CA 9527 A          JSR     @ERST           ;                       0459
F5CB 5118 A DELAY1:  LI      AC1,18          ;                       0460
F5CC 9522 A          JSR     @DELAY          ;                       0461
F5CD 156D A AGAIN:   JSR     INC1            ;                       0462
F5CE 1901 A          JMP     +2              ;                       0463
F5CF 1906 A          JMP     DEC             ;                       0464
F5D0 9520 A          JSR     @SHIFT3         ;                       0465
F5D1 C307 A          LD      AC0,007(AC3)    ;                       0466
F5D2 A91A A          AND     AC0,MA          ;                       0467
F5D3 B10F A          ST      AC0,@AD3        ;                       0468
F5D4 D307 A          ST      AC0,007(AC3)    ;                       0469
F5D5 19D5 A          JMP     LOOP2           ;                       0470
F5D6 156D A DEC:     JSR     PRINT           ;                       0471
F5D7 1573 A          JSR     ESUM            ;                       0472
```

```
F5D8 C307 A          LD      AC0,007(AC3)    ;                    0473
F5D9 A914 A          AND     AC0,MAS         ;                    0474
F5DA B108 A          ST      AC0,@AD3        ;                    0475
F5DB AF04 A          DSZ     004(AC3)        ;                    0476
F5DC 1901 A          JMP     +2              ;                    0477
F5DD 1952 A          JMP     END             ;                    0478
F5DE 7A04 A          A1SZ    AC2,4           ;                    0479
F5DF 19C1 A          JMP     LOOP1           ;                    0480
F5E0 194F A          JMP     END             ;                    0481
F5E1 0100 A AD1:     .WORD   00100           ;                    0482
F5E2 9000 A AD2:     .WORD   09000           ;                    0483
F5E3 A000 A AD3:     .WORD   0A000           ;                    0484
F5E4 4010 A D21:     .WORD   04010           ;                    0485
F5E5 4810 A D22:     .WORD   04810           ;                    0486
F5E6 5812 A D23:     .WORD   05812           ;                    0487
F5E7 5A12 A D24:     .WORD   05A12           ;                    0488
F5E8 5A1A A D25:     .WORD   05A1A           ;                    0489
F5E9 5A3A A D26:     .WORD   05A3A           ;                    0490
F5EA 1812 A D27:     .WORD   01812           ;                    0491
F5EB 1C12 A D28:     .WORD   01C12           ;                    0492
F5EC 0010 A M:       .WORD   00010           ;                    0493
F5ED FF10 A MA:      .WORD   0FF10           ;                    0494
F5EE FF0F A MAS:     .WORD   0FF0F           ;                    0495
F5EF F2FF A DELAY:   .WORD   0F2FF           ;                    0496
F5F0 F6AC A SHIFT1:  .WORD   0F6AC           ;                    0497
F5F1 F6A6 A SHIFT3:  .WORD   0F6A6           ;                    0498
F5F2 F694 A ERST:    .WORD   0F694           ;                    0499
F5F3 C1F0 A SM1:     LD      AC0,D21         ;                    0500
F5F4 B1ED A          ST      AC0,@AD2        ;                    0501
F5F5 C1EF A          LD      AC0,D22         ;                    0502
F5F6 B1EB A          ST      AC0,@AD2        ;                    0503
F5F7 C1EE A          LD      AC0,D23         ;                    0504
F5F8 B1E9 A          ST      AC0,@AD2        ;                    0505
F5F9 8000 A          RTS                     ;                    0506
F5FA B1E8 A SM2:     ST      AC0,@AD3        ;                    0507
F5FB A5F0 A          OR      AC0,M           ;                    0508
F5FC B1E6 A          ST      AC0,@AD3        ;                    0509
F5FD D307 A          ST      AC0,007(AC3)    ;                    0510
F5FE 5000 A          LI      AC0,0           ;                    0511
F5FF D300 A          ST      AC0,(AC3)       ;                    0512
F600 8000 A          RTS                     ;                    0513
F601 C1E8 A SVER:    LD      AC0,D27         ;                    0514
F602 B1DF A          ST      AC0,@AD2        ;                    0515
F603 C1E7 A          LD      AC0,D28         ;                    0516
F604 B1DD A          ST      AC0,@AD2        ;                    0517
F605 9561 A          JSR     @SENSE1         ;                    0518
F606 C1E3 A          LD      AC0,D27         ;                    0519
F607 B1DA A          ST      AC0,@AD2        ;                    0520
F608 C1DD A          LD      AC0,D23         ;                    0521
F609 B1D8 A          ST      AC0,@AD2        ;                    0522
F60A 8000 A          RTS                     ;                    0523
F60B 5000 A SUMV:    LI      AC0,0           ;                    0524
F60C CDD4 A          LD      AC3,AD1         ;                    0525
F60D D304 A          ST      AC0,004(AC3)    ;                    0526
F60E 15E4 A          JSR     SM1             ;                    0527
F60F C309 A LOOP1:   LD      AC0,009(AC3)    ;                    0528
F610 D301 A          ST      AC0,001(AC3)    ;                    0529
F611 C304 A          LD      AC0,004(AC3)    ;                    0530
F612 1573 A          JSR     SM3             ;                    0531
```

```
F613 C307 A LOOP2:  LD      AC0,007(AC3)  ;                    0532
F614 A700 A         OR      AC0,(AC3)     ;                    0533
F615 B1CD A         ST      AC0,@AD3      ;                    0534
F616 D307 A         ST      AC0,007(AC3)  ;                    0535
F617 15E9 A         JSR     SVER          ;                    0536
F618 C301 A         LD      AC0,001(AC3)  ;                    0537
F619 4304 A         BOC     BIT 0,+4      ;                    0538
F61A 1525 A         JSR     SENCH         ;                    0539
F61B 1903 A         JMP     +4            ;                    0540
F61C 1903 A         JMP     AGAIN         ;                    0541
F61D 1522 A         JSR     SENCH         ;                    0542
F61E 1901 A         JMP     +2            ;                    0543
F61F 95D2 A         JSR     @ERST         ;                    0544
F620 151A A AGAIN:  JSR     INC1          ;                    0545
F621 1901 A         JMP     +2            ;                    0546
F622 1906 A         JMP     DEC           ;                    0547
F623 95CC A         JSR     @SHIFT1       ;                    0548
F624 C307 A         LD      AC0,007(AC3)  ;                    0549
F625 A9C7 A         AND     AC0,MA        ;                    0550
F626 B1BC A         ST      AC0,@AD3      ;                    0551
F627 D307 A         ST      AC0,007(AC3)  ;                    0552
F628 19EA A         JMP     LOOP2         ;                    0553
F629 151A A DEC:    JSR     PRINT         ;                    0554
F62A 1520 A         JSR     ESUM          ;                    0555
F62B C307 A         LD      AC0,007(AC3)  ;                    0556
F62C A9C1 A         AND     AC0,MAS       ;                    0557
F62D B1B5 A         ST      AC0,@AD3      ;                    0558
F62E 1505 A         JSR     INC3          ;                    0559
F62F 19DF A         JMP     LOOP1         ;                    0560
F630 C1B4 A END:    LD      AC0,D22       ;                    0561
F631 B1B0 A         ST      AC0,@AD2      ;                    0562
F632 C1B1 A         LD      AC0,D21       ;                    0563
F633 9934 A         JMP     @FIN          ;                    0564
F634 5030 A INC3:   LI      AC0,30        ;                    0565
F635 8F04 A         ISZ     004(AC3)      ;                    0566
F636 F304 A         SKNE    AC0,004(AC3)  ;                    0567
F637 8001 A         RTS     +1            ;                    0568
F638 8000 A         RTS                   ;                    0569
F639 5010 A INC2:   LI      AC0,10        ;                    0570
F63A 1901 A         JMP     +2            ;                    0571
F63B 5008 A INC1:   LI      AC0,8         ;                    0572
F63C 8F00 A         ISZ     (AC3)         ;                    0573
F63D F300 A         SKNE    AC0,(AC3)     ;                    0574
F63E 8001 A         RTS     +1            ;                    0575
F63F 8000 A         RTS                   ;                    0576
F640 C305 A SENCH:  LD      AC0,005(AC3)  ;                    0577
F641 4301 A CHECK:  BOC     BIT0,+2       ;                    0578
F642 8000 A         RTS                   ;                    0579
F643 8001 A         RTS     +1            ;                    0580
F644 C30A A PRINT:  LD      AC0,00A(AC3)  ;                    0581
F645 4501 A         BOC     BIT<>0,+2     ;                    0582
F646 8001 A         RTS     +1            ;                    0583
F647 CD02 A         LD      AC3,AD5       ;                    0584
F648 1554 A         JSR     SENDA         ;                    0585
F649 8000 A         RTS                   ;                    0586
F64A F675 A AD5:    .WORD                 ;                    0587
F64B C1DF A ESUM:   LD      AC3,AD8       ;                    0588
F64C 1550 A         JSR     SENDA         ;                    0589
F64D 153D A         JSR     PRPT          ;                    0590
```

```
F64E 1906 A           JMP     EOUT
F64F CD1A A EPRD:     LD      AC3,AD7
F650 154C A           JSR     SENDA
F651 1559 A           JSR     PRPT
F652 CD19 A           LD      AC3,AD9
F653 1549 A           JSR     SENDA
F654 1902 A           JMP     ERPR
F655 CD13 A EOUT:     LD      AC3,AD6
F656 1546 A           JSR     SENDA
F657 6200 A ERPR:     PUSH    AC2
F658 CD88 A           LD      AC3,AD1
F659 C913 A           LD      AC2,ADA
F65A 7A01 A           AISZ    AC2,1
F65B C200 A           LD      AC0,(AC2)
F65C 1554 A           JSR     SENDN
F65D C200 A           LD      AC0,(AC2)
F65F 2C0R A           SHR     AC0,4,0
F65F 950F A           JSR     @SEND
F660 502C A           LI      AC0,2C
F661 950D A           JSR     @SEND
F662 AF0A A           DSZ     00A(AC3)
F663 19F6 A           JMP     ERPR+3
F664 9509 A           JSR     @CRLF
F665 6600 A           PULL    AC2
F666 8000 A           RTS
F667 F6BA A SENSE1:   .WORD   0F6BA
F668 F6C6A   FIN:     .WORD   0F6C6
F669 F67C A AD6:      .WORD   0F67C
F66A F681 A AD7:      .WORD   0F681
F66B F670 A AD8:      .WORD   0F670
F66C F684 A AD9:      .WORD   0F684
F66D 0104 A ADA:      .WORD   0010A
F66E F26B A CRLF:     .WORD   0F26B
F66F F319 A SEND:     .WORD   0F319
F670 5553 A           .WORD   05553
F671 204D A           .WORD   0204D
F672 FFFF A           .WORD   0FFFF
F673 2D50 A           .WORD   02D50
F674 FFFF A           .WORD   0FFFF
F675 4520 A           .WORD   04520
F676 5252 A           .WORD   05252
F677 524F A           .WORD   0524F
F678 2A20 A           .WORD   02A20
F679 2A2A A           .WORD   02A2A
F67A 0020 A           .WORD   00020
F67B FFFF A           .WORD   0FFFF
F67C 4F20 A           .WORD   04F20
F67D 5455 A           .WORD   05455
F67E 5550 A           .WORD   05550
F67F 2054 A           .WORD   02054
F680 FFFF A           .WORD   0FFFF
F681 5250 A           .WORD   05250
F682 2044 A           .WORD   02044
F683 FFFF A           .WORD   0FFFF
F684 4920 A           .WORD   04920
F685 FFFF A           .WORD   0FFFF
F686 2810 A SM3:      SHL     AC0,8,0
F687 9501 A           JSR     @SM2
F688 8000 A           RTS
```

```
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
```

```
F689 F5FA A SM2:    .WORD   0F5FA           ;                       0650
F68A 0000 A         .WORD   00000           ;                       0651
F68B CD07 A PRPT:   LD      AC3,A           ;                       0652
F68C 1510 A         JSR     SENDA           ;                       0653
F68D C200 A         LD      AC0,(AC2)       ;                       0654
F68E 2C08 A         SHL     AC0,8,0         ;                       0655
F68F 1521 A         JSR     SENDN           ;                       0656
F690 C200 A         LD      AC0,(AC2)       ;                       0657
F691 151F A         JSR     SENDN           ;                       0658
F692 8000 A         RTS                     ;                       0659
F693 F673 A A:      .WORD   0F673           ;                       0660
F694 6200 A ERST:   PUSH    AC2             ;                       0661
F695 C9D7 A         LD      AC2,ADA         ;                       0662
F696 8F0A A         ISZ     00A(AC3)        ;                       0663
F697 EB0A A         ADD     AC2,00A(AC3)    ;                       0664
F698 C300 A         LD      AC0,(AC3)       ;                       0665
F699 D2C0 A         ST      AC0,(AC2)       ;                       0666
F69A 5D80 A         RCPY    AC2,AC1         ;                       0667
F69B 6600 A         PULL    AC2             ;                       0668
F69C 8000 A         RTS                     ;                       0669
F69D C300 A SENDA:  LD      AC0,(AC3)       ;                       0670
F69E 4B05 A         BOC     AC0<0,+6        ;                       0671
F69F 95CF A         JSR     @SEND           ;                       0672
F6A0 2C10 A         SHR     AC0,8,0         ;                       0673
F6A1 95CD A         JSR     @SEND           ;                       0674
F6A2 7F01 A         AISZ    AC3,1           ;                       0675
F6A3 19F6 A         JMP     SENDA           ;                       0676
F6A4 8000 A         RTS                     ;                       0677
F6A5 F319 A SEND:   .WORD   0F319           ;                       0678
F6A6 C303 A SHIFT3: LD      AC0,003(AC3)    ;                       0679
F6A7 2C02 A         SHR     AC0,1,0         ;                       0680
F6A8 D303 A         ST      AC0,003(AC3)    ;                       0681
F6A9 C302 A SHIFT2: LD      AC0,002(AC3)    ;                       0682
F6AA 2C02 A         SHR     AC0,1,0         ;                       0683
F6AB D302 A         ST      AC0,002(AC3)    ;                       0684
F6AC C301 A SHIFT1: LD      AC0,001(AC3)    ;                       0685
F6AD 2C02 A         SHR     AC0,1,0         ;                       0686
F6AE D301 A         ST      AC0,001(AC3)    ;                       0687
F6AF 8000 A         RTS                     ;                       0688
F6B0 5C00 A         NOP                     ;                       0689
F6B1 A906 A SENDN:  AND     AC0,MASK        ;                       0690
F6B2 7830 A         AISZ    AC0,30          ;                       0691
F6B3 9D05 A         SKG     AC0,ALFA        ;                       0692
F6B4 1901 A         JMP     +2              ;                       0693
F6B5 7807 A         AISZ    AC0,7           ;                       0694
F6B6 95B8 A         JSR     @SEND           ;                       0695
F6B7 8000 A         RTS                     ;                       0696
F6B8 000F A MASK :  .WORD   0000F           ;                       0697
F6B9 0039 A ALFA:   .WORD   00039           ;                       0698
F6BA A109 A SENSE1: LD      AC0,@AD4        ;                       0699
F6BB A909 A         AND     AC0,MK          ;                       0700
F6BC D305 A .       ST      AC0,005(AC3)    ;                       0701
F6BD 8000 A         RTS                     ;                       0702
F6BE A105 A SENSE2: LD      AC0,@AD4        ;                       0703
F6BF A905 A         AND     AC0,MK          ;                       0704
F6C0 D306 A         ST      AC0,006(AC3)    ;                       0705
F6C1 8000 A         RTS                     ;                       0706
F6C2 9000 A AD2:    .WORD   09000           ;                       0707
F6C3 A000 A AD3:    .WORD   0A000           ;                       0708
```

```
F6C4 C000 A AD4:    .WORD   0C000           ;                       0709
F6C5 0001 A MK:     .WORD   00001           ;                       0710
F6C6 B1FB A FIN:    ST      AC0,@AD2        ;                       0711
F6C7 5000 A         LI      AC0,0           ;                       0712
F6C8 B1F9 A         ST      AC0,@AD2        ;                       0713
F6C9 B1F9 A         ST      AC0,@AD3        ;                       0714
F6CA 8000 A         RTS                     ;                       0715
F6CB D307 A STV:    ST      AC0,007(AC3)    ;                       0716
F6CC B1F6 A         ST      AC0,@AD3        ;                       0717
F6CD 8000 A         RTS                     ;                       0718
```

APPENDIX E

This appendix provides the listing of the modified version of the minimization program |10|. An asterisk (*) appears beside each modified statement.

```
C                                                                       0001
C                                                                       0002
C.......                        A PROGRAM FOR:                          0003
C.......  THE MINIMIZATION OF LOGIC FUNCTIONS AS APPLIED TO PLA-S       0004
C                                                                       0005
C.....        *****   MAIN PROGRAM  *****                               0006
C                                                                       0007
C                                                                       0008
*       INTEGER*2 MCVR(128,70),MVR(70),MT(128,7),JC(512,3),IC(512,3),   0009
*      1MV(128,4),MW(256),MK(256),M2(8),MST(46,10),F(8),KV(8)           0010
*       WRITE(7,1)                                                      0011
*     1 FORMAT(/,5X,'ENTER; NO.OF EXAMPLES')                            0012
*       READ(5,*) I                                                     0013
*  C.....TO READ DATA .                                                 0014
*       DO 10 J=1,I                                                     0015
*       DO 25 K=1,8                                                     0016
*    25 F(K)=0                                                          0017
*       WRITE(7,2)                                                      0018
*     2 FORMAT(/,5X,'ENTER; NO.OF VARIABLES, NO.OF FUNCTIONS, MAX NO.OF AND0019
*      1C-GATES')                                                       0020
*       READ(5,*) NV,NF,NMX                                             0021
*       WRITE(7,64)                                                     0022
*    64 FORMAT(/,5X,'ENTER; 1 IF OUTPUT FUNCTION TO BE INVERTED, 0 IF NOT 0023
*      1,FOR YOUR FUNCTIONS, IN SEQUENCE')                              0024
*       READ(5,*) (F(K),K=1,NF)                                         0025
*       KVN=0                                                           0026
*       DO 425 KW=1,NF                                                  0027
*       IF(F(KW).LT.1)  GO TO 425                                       0028
*       KVN=KVN+1                                                       0029
*       KV(KVN)= KW                                                     0030
*   425 CONTINUE                                                        0031
*       NN=2**NV                                                        0032
        N2N=2*NN                                                        0033
        NN2=NN/2                                                        0034
        CALL MAIN1(MC,MK,MVR,MT,JC,MW,MK,M2,NV,NF,NMX,NMC,NMW,IC,IMV,0035
       1,MST,KVN,KV)
*       IF(LNP1.GT.46)   WRITE(7,6) LNPI                                0036
*       L=0                                                             0037
*       LL=0                                                            0038
*       DO 55 JK=1,NF                                                   0039
*       JK1=JK-1                                                        0040
*       L=L+F(JK)*2**JK1                                                0041
*       LL=2**JK1+LL                                                    0042
*    55 CONTINUE                                                        0043
*       IF(L.EQ.0)   GO TO 155                                          0044
*       L=1+2*L                                                         0045
*   155 F(1)=LL                                                         0046
*       F(2)=L                                                          0047
*       F(3)=LNPI                                                       0048
*       WRITE(6,56) (F(I7),I7=1,3)                                      0049
*    56 FORMAT(//,10X,'AMD4',3(',',Z4))                                 0050
*       ML=2**16                                                        0051
*       ML1=ML-2**NV                                                    0052
*       DO 57 JL=1,LNPI                                                 0053
*       IL=0                                                            0054
*       DO 58 LJ=3,10                                                   0055
*       IF(MST(JL,LJ).EQ.0)  GO TO 59                                   0056
*       ML=MST(JL,LJ)-1                                                 0057
*       IL=2**ML+IL                                                     0058
*                                                                       0059
```

```
*    58  CONTINUE                                                                0060
*    59  MST(JL,4)=IL                                                           0061
*        MST(JL,3)=ML1+MST(JL,2)                                                 0062
*        MST(JL,2)=MST(JL,1)                                                     0063
*        MST(JL,1)=JL-1                                                          0064
*    57  CONTINUE                                                               0065
*        J3=64                                                                   0066
*        DO 60 KL=1,LNPI,3                                                       0067
*        KL1=KL+2                                                                0068
*        WRITE(6,3) J3,((MST(J1,J2),J2=1,4),J1=KL,KL1)                           0069
*        IF((KL+5).GE.LNPI)   GO TO 61                                           0070
*    60  J3=J3+12                                                               0071
*        GO TO 10                                                                0072
*    61  J3=J3+12                                                               0073
*        KS=KL1+1                                                                0074
*        IF((KL+3).EQ.LNPI) WRITE(6,4) J3,(MST(KS,J2),J2=1,4)                    0075
*        IF((KL+4).EQ.LNPI) WRITE(6,5) J3,((MST(J1,J2),J2=1,4),J1=KS,LNPI)       0076
*        IF((KL+5).EQ.LNPI) WRITE(6,3) J3,((MST(J1,J2),J2=1,4),J1=KS,LNPI)       0077
*      3 FORMAT(//,10X,'AM',Z2,12('*',Z4))                                       0078
*      4 FORMAT(//,10X,'AM',Z2,4('*',Z4))                                        0079
*      5 FORMAT(//,10X,'AM',Z2,8('*',Z4))                                        0080
*      6 FORMAT(/,5X,'IF YOU WANT TO USE THE PLA, YOU SHOULD SPLIT YOUR FUNC0081
*       1CTIONS, BECAUSE THE PLA HAS ONLY 48 AND-GATES.THE NUMBER OF GATES 0082
*       1FOR THIS EXAMPLE =',I5)                                                 0083
     10 CONTINUE                                                                0084
        STOP                                                                    0085
        END                                                                     0086
*       SUBROUTINE MINIM(MCVR,MVK,MT,JC,IC,MV,MW,MK,M2,                         0087
*      1NV,NF,NN,N2N,NN2,NMX,LNPI,MST,KVN,KV)                                   0088
*       INTEGER*2 MT(NN,NF),MCVR(NN,NN2),MV(NN2,2),MVK(NN2),MST(48,10)          0089
        INTEGER*2 JC(N2N,3),IC(N2N,3),MW(NN),MK(NN),M2(NV),KV(8)                0090
        WRITE(6,100)NV,NF,NMX                                                   0091
    100 FORMAT(1H1,//,20X,21('* '),//,37X,'DATA',//,10X,                        0092
       1'NO. OF VARIABLES=',I2,//,10X,'NO. OF FUNCTIONS=',I2,//,10X,            0093
       2 'MAX. ROSS.ELL NO. OF AND- GATES=',I3,/)                               0094
*       IF(KVN.GT.0) WRITE(6,556) (KV(KV1),KV1=1,KVN)                           0095
*   556 FORMAT(10X,'INVERTED FUNCTIONS:',8(I2,2X))                              0096
        CALL MTFIL(NN,NF,MT,MW,L)                                               0097
        IF(L.EQ.0) RETURN                                                       0098
C.......TO FORM PRIME IMPLICANTS; FIRST FOR SINGLE FUNCTIONS                    0099
        CALL FNDNG(NN,NF,MT,NG)                                                 0100
        WRITE(6,109)                                                            0101
    109 FORMAT(1H1,//,20X,21('* '),/,35X,'RESULTS:',//)                         0102
        WRITE(6,108)NG                                                          0103
    108 FORMAT(//,10X,'NO. OF REQUIRED AND-GATES BEFORE MINIMIZATION=',I5)      0104
        IF(NG.LE.NMX) GO TO 99                                                  0105
        LNPI=0                                                                  0106
     20 DO 11 K=1,NF                                                            0107
        CALL MWPMT(NN,NF,NV,K,MT,MW,1,0)                                        0108
        CALL CNTNX(NN,MW,NX)                                                    0109
        IF(NX.EQ.0) GO TO 11                                                    0110
        WRITE(6,104)K                                                           0111
    104 FORMAT(10X,5('* '),'PRIME IMPLICANTS COVERING FUNCTION-',I2,/)          0112
        CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                           0113
        CALL MWPMT(NN,NF,NV,K,MT,MW,1,1)                                        0114
*       CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K,0,0,0,0,0,0,0,MS0115
*      1T)                                                                      0116
        CALL MODMT(NN,NF,K,MT,MK)                                               0117
        CALL FNDNG(NN,NF,MT,NG)                                                 0118
```

```
      IF(NG.EQ.0) GO TO 98                                                  0119
      NG=NG+LNPI                                                            0120
      IF(NG.LE.NMX) GO TO 99                                                0121
   11 CONTINUE                                                              0122
C.....TO FORM PRIME IMPLICANTS FOR MULTIPLE FUNCTIONS                       0123
C.....TO FORM PRIME IMPLICANTS FOR PRODUCT OF 2-FUNCTIONS                   0124
      IF(NF.LT.2) GO TO 97                                                  0125
      WRITE(6,105)                                                          0126
      N2=NF-1                                                               0127
      DO 12 K1=1,N2                                                         0128
      KK2=K1+1                                                              0129
      DO 12 K2=KK2,NF                                                       0130
      CALL MW2MT(NN,NF,NV,K1,K2,MT,MW,2,0)                                  0131
      CALL CNTNX(NN,MW,NX)                                                  0132
      IF(NX.EQ.0) GO TO 12                                                  0133
      WRITE(6,107)K1,K2                                                     0134
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                         0135
*     CALL MW2MT(NN,NF,NV,K1,K2,MT,MW,2,1)                                  0136
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,0,0,0,0,0,0,0,0137
*    1MST)                                                                  0138
      CALL MODMT(NN,NF,K1,MT,MK)                                            0139
      CALL MODMT(NN,NF,K2,MT,MK)                                            0140
      CALL FNDNG(NN,NF,MT,NG)                                               0141
      IF(NG.EQ.0) GO TO 98                                                  0142
      NG=NG+LNPI                                                            0143
      IF(NG.LE.NMX) GO TO 99                                                0144
   12 CONTINUE                                                              0145
  105 FORMAT(1H1,//,10X,'TO FIND PI-S FOR PRODUCT OF 2-FUNCTIONS:',//)      0146
  107 FORMAT(10X,5('* '),'PI-S COVERING PROD. FUNS. : F-',2(I2,',',),/)     0147
C.....TO FORM PRIMEIMPLICANTS FOR PRODUCT OF 3-FUNCTIONS                    0148
      IF(NF.LT.3) GO TO 97                                                  0149
      WRITE(6,110)                                                          0150
      N3=NF-2                                                               0151
      DO 13 K1=1,N3                                                         0152
      KK2=K1+1                                                              0153
      DO 13 K2=KK2,N2                                                       0154
      KK3=K2+1                                                              0155
      DO 13 K3=KK3,NF                                                       0156
      CALL MW3MT(NN,NF,NV,K1,K2,K3,MT,MW,3,0)                               0157
      CALL CNTNX(NN,MW,NX)                                                  0158
      IF(NX.EQ.0) GO TO 13                                                  0159
      WRITE(6,112)K1,K2,K3                                                  0160
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                         0161
      CALL MW3MT(NN,NF,NV,K1,K2,K3,MT,MW,3,1)                               0162
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,0,0,0,0,00163
*    1,MST)                                                                 0164
      CALL MODMT(NN,NF,K1,MT,MK)                                            0165
      CALL MODMT(NN,NF,K2,MT,MK)                                            0166
      CALL MODMT(NN,NF,K3,MT,MK)                                            0167
      CALL FNDNG(NN,NF,MT,NG)                                               0168
      IF(NG.EQ.0) GO TO 98                                                  0169
      NG=NG+LNPI                                                            0170
      IF(NG.LE.NMX) GO TO 99                                                0171
   13 CONTINUE                                                              0172
  110 FORMAT(1H1,//,10X,'TO FIND PI-S FOR PRODUCT OF 3-FUNCTIONS:',//)      0173
  112 FORMAT(10X,5('* '),'PI-S COVERING PROD. FUNS. : F-',3(I2,',',),/)     0174
C.....TO FORM PRIMEIMPLICANTS FOR PRODUCT OF 4-FUNCTIONS                    0175
      IF(NF.LT.4) GO TO 97                                                  0176
      WRITE(6,120)                                                          0177
```

```
      N4=NF-3                                                           0178
      DO 14 K1=1,N4                                                     0179
      KK2=K1+1                                                          0180
      DO 14 K2=KK2,N3                                                   0181
      KK3=K2+1                                                          0182
      DO 14 K3=KK3,N2                                                   0183
      KK4=K3+1                                                          0184
      DO 14 K4=KK4,NF                                                   0185
      CALL MW4MT(NN,NF,NV,K1,K2,K3,K4,MT,MW,4,0)                        0186
      CALL CNTNX(NN,MW,NX)                                              0187
      IF(NX.EQ.0) GO TO 14                                              0188
      WRITE(6,122)K1,K2,K3,K4                                           0189
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                     0190
      CALL MW4MT(NN,NF,NV,K1,K2,K3,K4,MT,MW,4,1)                        0191
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,K4,0,0,0,0192
*    1C,MST)                                                            0193
      CALL MCDMT(NN,NF,K1,MT,MK)                                        0194
      CALL MCDMT(NN,NF,K2,MT,MK)                                        0195
      CALL MCDMT(NN,NF,K3,MT,MK)                                        0196
      CALL MCDMT(NN,NF,K4,MT,MK)                                        0197
      CALL FNDNC(NN,NF,MT,NC)                                           0198
      IF(NC.EQ.0) GO TO 98                                              0199
      NC=NC+LNPI                                                        0200
      IF(NC.LE.NMX) GO TO 99                                            0201
   14 CONTINUE                                                          0202
  120 FORMAT(1H1,//,1C,'TO FIND PI-S FOR PRODUCT OF 4-FUNCTIONS:',//)   0203
  122 FORMAT(10X,5('* '),'PI-S COVERING PROD. FUNS. : F=',4(I2,','),/)  0204
C.........TO FORM PRIMEIMPLICANTS FOR PRODUCT OF 5-FUNCTIONS            0205
      IF(NF.LT.5) GO TO 97                                              0206
      WRITE(6,130)                                                      0207
      N5=NF-4                                                           0208
      DO 15 K1=1,N5                                                     0209
      KK2=K1+1                                                          0210
      DO 15 K2=KK2,N4                                                   0211
      KK3=K2+1                                                          0212
      DO 15 K3=KK3,N3                                                   0213
      KK4=K3+1                                                          0214
      DO 15 K4=KK4,N2                                                   0215
      KK5=K4+1                                                          0216
      DO 15 K5=KK5,NF                                                   0217
      CALL MW5MT(NN,NF,NV,K1,K2,K3,K4,K5,MT,MW,5,0)                     0218
      CALL CNTNX(NN,MW,NX)                                              0219
      IF(NX.EQ.0) GO TO 15                                              0220
      WRITE(6,132)K1,K2,K3,K4,K5                                        0221
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                     0222
      CALL MW5MT(NN,NF,NV,K1,K2,K3,K4,K5,MT,MW,5,1)                     0223
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,K4,K5,C,00224
*    1,0,MST)                                                           0225
      CALL MCDMT(NN,NF,K1,MT,MK)                                        0226
      CALL MCDMT(NN,NF,K2,MT,MK)                                        0227
      CALL MCDMT(NN,NF,K3,MT,MK)                                        0228
      CALL MCDMT(NN,NF,K4,MT,MK)                                        0229
      CALL MCDMT(NN,NF,K5,MT,MK)                                        0230
      CALL FNDNC(NN,NF,MT,NC)                                           0231
      IF(NC.EQ.0) GO TO 98                                              0232
      NC=NC+LNPI                                                        0233
      IF(NC.LE.NMX) GO TO 99                                            0234
   15 CONTINUE                                                          0235
  130 FORMAT(1H1,//,10X,'TO FIND PI-S FOR PRODUCT OF 5-FUNCTIONS:',//)  0236
```

```
  132 FORMAT(10X,5('* '),'P1-S COVERING PROD. FUNS. : F-',2(4(I2,','),     0237
     1/,51X),/)                                                           0238
C.........TO FORM PRIMEIMPLICANTS FOR PRODUCT OF 6-FUNCTIONS             0239
      IF(NF.LT.6) GO TO 97                                               0240
      WRITE(6,140)                                                       0241
      N6=NF-5                                                            0242
      DO 16 K1=1,N6                                                      0243
      KK2=K1+1                                                           0244
      DO 16 K2=KK2,N5                                                    0245
      KK3=K2+1                                                           0246
      DO 16 K3=KK3,N4                                                    0247
      KK4=K3+1                                                           0248
      DO 16 K4=KK4,N3                                                    0249
      KK5=K4+1                                                           0250
      DO 16 K5=KK5,N2                                                    0251
      KK6=K5+1                                                           0252
      DO 16 K6=KK6,NF                                                    0253
      CALL MW6MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,MT,MW,6,0)                   0254
      CALL CNTNX(NN,MW,NX)                                               0255
      IF(NX.EQ.0) GO TO 16                                               0256
      WRITE(6,142)K1,K2,K3,K4,K5 K6                                      0257
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                      0258
      CALL MW6MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,MT,MW,6,1)                   0259
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,K4,K5,K6,0260
*    1C,0,MST)                                                           0261
      CALL MCDMT(NN,NF,K1,MT,MK)                                         0262
      CALL MCDMT(NN,NF,K2,MT,MK)                                         0263
      CALL MCDMT(NN,NF,K3,MT,MK)                                         0264
      CALL MCDMT(NN,NF,K4,MT,MK)                                         0265
      CALL MCDMT(NN,NF,K5,MT,MK)                                         0266
      CALL MCDMT(NN,NF,K6,MT,MK)                                         0267
      CALL SNPNC(NN,NF,MT,NC)                                            0268
      IF(NG.EQ.0) GO TO 98                                               0269
      NG=NG+LNPI                                                         0270
      IF(NG.EQ.MAXG) GO TO 99                                            0271
   16 CONTINUE                                                           0272
  140 FORMAT(1H1,//,10X,'TO FIND P1-S FOR PRODUCT OF 6-FUNCTIONS:',//)   0273
  142 FORMAT(10X,5('* '),'P1-S COVERING PROD. FUNS. : F-',2(4(I2,','),   0274
     1/,51X),/)                                                           0275
C.........TO FORM PRIMEIMPLICANTS FOR PRODUCT OF 7-FUNCTIONS             0276
      IF(NF.LT.7) GO TO 97                                               0277
      WRITE(6,150)                                                       0278
      N7=NF-6                                                            0279
      DO 17 K1=1,N7                                                      0280
      KK2=K1+1                                                           0281
      DO 17 K2=KK2,N6                                                    0282
      KK3=K2+1                                                           0283
      DO 17 K3=KK3,N5                                                    0284
      KK4=K3+1                                                           0285
      DO 17 K4=KK4,N4                                                    0286
      KK5=K4+1                                                           0287
      DO 17 K5=KK5,N3                                                    0288
      KK6=K5+1                                                           0289
      DO 17 K6=KK6,N2                                                    0290
      KK7=K6+1                                                           0291
      DO 17 K7=KK7,NF                                                    0292
      CALL MW7MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,K7,MT,MW,7,0)                0293
      CALL CNTNX(NN,MW,NX)                                               0294
      IF(NX.EQ.0) GO TO 17                                               0295
```

```
      WRITE(6,152)K1,K2,K3,K4,K5 K6,K7                                  0296
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                     0297
*     CALL MW7MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,K7,MT,MW,7,1)               0298
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,K4,K5,K6,0299
*    1K7,0,MST)                                                         0300
      CALL MODMT(NN,NF,K1,MT,MK)                                        0301
      CALL MODMT(NN,NF,K2,MT,MK)                                        0302
      CALL MODMT(NN,NF,K3,MT,MK)                                        0303
      CALL MODMT(NN,NF,K4,MT,MK)                                        0304
      CALL MODMT(NN,NF,K5,MT,MK)                                        0305
      CALL MODMT(NN,NF,K6,MT,MK)                                        0306
      CALL MODMT(NN,NF,K7,MT,MK)                                        0307
      CALL FNDNC(NN,NF,MT,NC)                                           0308
      IF(NC.EQ.0) GO TO 98                                              0309
      NC=NC+LNPI                                                        0310
      IF(NC.LE.NMX) GO TO 99                                            0311
   17 CONTINUE                                                          0312
  150 FORMAT(1H1,//,10X,'TO FIND PI-S FOR PRODUCT OF 7-FUNCTIONS:',//)  0313
  152 FORMAT(10X,5('** '),'PI-S COVERING PROD. FUNS. : F-',2(4(I2,','),0314
     1/,51X),/)                                                         0315
C.....TO FORM PRIMEIMPLICANTS FOR PRODUCT OF 8-FUNCCIONS                0316
      IF(NF.LT.8) GO TO 97                                              0317
      WRITE(6,160)                                                      0318
      N8=NF-7                                                           0319
      DO 18 K1=1,N8                                                     0320
      KK2=K1+1                                                          0321
      DO 18 K2=KK2,N7                                                   0322
      KK3=K2+1                                                          0323
      DO 18 K3=KK3,N6                                                   0324
      KK4=K3+1                                                          0325
      DO 18 K4=KK4,N5                                                   0326
      KK5=K4+1                                                          0327
      DO 18 K5=KK5,N4                                                   0328
      KK6=K5+1                                                          0329
      DO 18 K6=KK6,N3                                                   0330
      KK7=K6+1                                                          0331
      DO 18 K7=KK7,N2                                                   0332
      KK8=K7+1                                                          0333
      DO 18 K8=KK8,NF                                                   0334
      CALL MW8MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,K7,K8,MT,MW,8,0)            0335
      CALL CNTNX(NN,MW,NX)                                              0336
      IF(NX.EQ.0) GO TO 18                                              0337
      WRITE(6,162)K1,K2,K3,K4,K5 K6,K7,K8                               0338
      CALL MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                     0339
*     CALL MW8MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,K7,K8,MT,MW,8,1)            0340
*     CALL SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,K4,K5,K6,0341
*    1K7,K8,MST)                                                        0342
      CALL MODMT(NN,NF,K1,MT,MK)                                        0343
      CALL MODMT(NN,NF,K2,MT,MK)                                        0344
      CALL MODMT(NN,NF,K3,MT,MK)                                        0345
      CALL MODMT(NN,NF,K4,MT,MK)                                        0346
      CALL MODMT(NN,NF,K5,MT,MK)                                        0347
      CALL MODMT(NN,NF,K6,MT,MK)                                        0348
      CALL MODMT(NN,NF,K7,MT,MK)                                        0349
      CALL MODMT(NN,NF,K8,MT,MK)                                        0350
      CALL FNDNC(NN,NF,MT,NC)                                           0351
      IF(NC.EQ.0) GO TO 98                                              0352
      NC=NC+LNPI                                                        0353
      IF(NC.LE.NMX) GO TO 99                                            0354
```

```
     18 CONTINUE                                                        0355
    160 FORMAT(1H1,//,10X,'TO FIND PI-S FOR PRODUCT OF 8-FUNCTIONS:',//) 0356
    162 FORMAT(10X,5('* '),'PI-S COVERING PROD. FUNS. : F-',2(4(I2,',',') 0357
       1,,3X),/)                                                        0358
     97 WRITE(6,113)                                                    0359
    113 FORMAT (/,20X,21('* '),/,10X,'THE PROCESS IS REPEATED ONCE MORE') 0360
        CALL ARANG(NN,NF,MT)                                            0361
        GO TO 20                                                        0362
     98 WRITE(6,114)LNPI                                                0363
    114 FORMAT (/,10X,'NO FURTHER MINIMIZATION COULD BE APPLIED',//,    0364
       110X,'NO. OF AND-GATES=',I4,/,20X,21('- '))                      0365
        RETURN                                                          0366
     99 WRITE(6,200)NG                                                  0367
    200 FORMAT(10X,'NO. OF AND-GATES=',I4,/)                            0368
        CALL ARANG(NN,NF,MT)                                            0369
        DO 555 I=1,NN                                                   0370
        DO 555 J=1,NF                                                   0371
        IF(MT(I,J).EQ.2)MT(I,J)=1                                       0372
    555 CONTINUE                                                        0373
        WRITE(6,101)((MT(I,J),I=1,NN),J=1,NF)                           0374
    101 FORMAT(16(3X,'MT',3X,16I5,/),/)                                 0375
        RETURN                                                          0376
        END                                                             0377
C........SUBROUTINE TO COUNT NO. OF EXCLUSIVE MINTERMS USING MW(I) .    0378
C........NX= NO. OF EXCLUSIVE MINTERMS .                                0379
      SUBROUTINE CNTNX(NN,MW,NX)                                        0380
*     INTEGER*2 MW(NN)                                                  0381
      NX=0                                                              0382
      DO 20 I=1,NN                                                      0383
      IF(MW(I).EQ.1) GO TO 10                                           0384
      GO TO 20                                                          0385
   10 NX=NX+1                                                           0386
   20 CONTINUE                                                          0387
      RETURN                                                            0388
      END                                                               0389
C........SUBROUTINE TO FIND THE BINERY EQUIVALENT:M2(NV) OF THE         0390
C........DECIMAL NUMBER:M .                                             0391
      SUBROUTINE GENM2(NV,M,M2)                                         0392
*     INTEGER*2 M2(NV)                                                  0393
      DO 10 I=1,NV                                                      0394
      N=2**(NV-I)                                                       0395
      J=NV+1-I                                                          0396
      IF(M-N)11,12,12                                                   0397
   11 M2(J)=0                                                           0398
      GO TO 10                                                          0399
   12 M2(J)=1                                                           0400
      M=M-N                                                             0401
   10 CONTINUE                                                          0402
      RETURN                                                            0403
      END                                                               0404
C . ... .SUBROUTINE TO FIND 1-CUBES FROM 0-CUBES, FIND 0-CUBE PI-S.     0405
C........NO. OF 1-CUBES = IR                                           0406
C .......NO. OF 0-CUBES PI-S = N                                       0407
      SUBROUTINE GEN1C(NN,N2N,NN2,NV,MW,MK,M2,IC,IR,MV,N)               0408
*     INTEGER*2 MW(NN),MK(NN),M2(NV),IC(N2N,3),MV(NN2,2)                0409
      DO 29 I=1,NN                                                      0410
   29 MK(I)=0                                                           0411
      IR=0                                                              0412
      DO 10 I=1,NN                                                      0413
```

```
      IF(MW(I).EQ.0) GO TO 10                                          0414
C.....AT THIS STEP MW(I)=1 OR 2 .                                      0415
      M=I-1                                                            0416
      CALL GENMI(NV,M,M2)                                              0417
      DO 11 J=1,NV                                                     0418
      IF(M2(J).EQ.1) GO TO 11                                          0419
C.....M2(I)=0 AT THIS STEP                                             0420
      K=I+2**(J-1)                                                     0421
      IF(MW(K).EQ.0) GO TO 11                                          0422
C.....AT THIS STEP:2 MW(K)=1 OR 2; MW(I)=1 OR 2                        0423
      IF(MW(I).EQ.1) GO TO 12                                          0424
C.....AT THIS STEP, MW(I)=2;MW(K)=1 OR 2                               0425
      IF(MW(K).EQ.2) GO TO 13                                          0426
C.....AT THIS STEP,  MW(I)=1&MW(K)=1OR2;OR MW(I)=2&MW(K)=1             0427
   12 IR=IR+1                                                          0428
      IC(IR,1)=I-1                                                     0429
      IC(IR,2)=K-1                                                     0430
      IC(IR,3)=0                                                       0431
      MK(I)=1                                                          0432
      MK(K)=1                                                          0433
      GO TO 11                                                         0434
C. . . . FOR THE FOLLOWING : BOTH MW(I)=2,AND MW(K)=2.                 0435
   13 IR=IR+1                                                          0436
      IC(IR,1)=I-1                                                     0437
      IC(IR,2)=K-1                                                     0438
      IC(IR,3)=2                                                       0439
      MK(I)=1                                                          0440
      MK(K)=1                                                          0441
   11 CONTINUE                                                         0442
   10 CONTINUE                                                         0443
      N=0                                                              0444
      DO 15 I=1,NN                                                     0445
      IF(MW(I).NE.1) GO TO 15                                          0446
      IF(MK(I).EQ.1) GO TO 15                                          0447
      N=N+1                                                            0448
      MV(N,1)=I-1                                                      0449
      MV(N,2)=0                                                        0450
   15 CONTINUE                                                         0451
      RETURN                                                           0452
      END                                                              0453
C.....TO FIND (I+1)-CUBES FROM I-CUBES.                                0454
C.....USE JC(I,1) TO FORMIC(I,1) & FILL MV(I,1).                       0455
C.....NO. OF ROWS IN JC=JR (JR=IR OF THE PREVICUS STEP)  (INPUT).      0456
C.....NO. OF ROWS IN MV(I,1)=M  (INPUT).                               0457
C.....MV(I,1) TO BE FILLED STARTING FROM ROW 'M+1' .                   0458
C.....NO. OF ROWS IN IC=IR  (OUTPUT).                                  0459
C.....NO. OF ROWS IN MV AFTER ALL PI'S ARE FOUND=N                     0460
C.....JR=IR ALWAYS BEFORE CALLING;ALSO M=N                             0461
C.....JR.GT.0 CHEK BEFORE CALLING.                                     0462
      SUBROUTINE ICFJC(NN,N2N,NN2,NV,M2,JC,JR,IC,IR,M,MV,N)            0463
*     INTEGER*2 JC(N2N,3),IC(N2N,3),MV(NN2,2),M2(NV)                   0464
      IR=0                                                             0465
      IF(JR.EQ.1) GO TO 25                                             0466
      II=JR-1                                                          0467
      DO 10 I=1,II                                                     0468
      J=I                                                              0469
   11 J=J+1                                                            0470
      IF(J.GT.JR) GO TO 10                                             0471
      IF(JC(I,2).EQ.JC(J,2)) GO TO 13                                  0472
```

```
      GO TO 11
   12 L=JC(J,1)-JC(I,1)                                                  0473
      DO 12 K=1,NV                                                       0474
      IF(L.EQ.2**(K-1)) GO TO 16                                         0475
   12 CONTINUE                                                           0476
      GO TO 11                                                           0477
   16 I1=JC(I,1)                                                         0478
      CALL GENM2(NV,I1,M2)                                               0479
      IF(M2(K).EQ.0) GO TO 18                                            0480
      GO TO 11                                                           0481
   18 L1=JC(I,2)+L                                                       0482
      IF(IR.EQ.0) GO TO 17                                               0483
C. . . . TO CHECK IF THE CUBE HAS ALREADY BEEN FOUND.                    0484
      DO 19 L2=1,IR                                                      0485
      IF(IC(L2,2).NE.L1) GO TO 19                                        0486
      IF(IC(L2,1).EQ.JC(I,1)) GO TO 15                                   0487
      GO TO 19                                                           0488
C. . . . THE I-CUBE IS ALREADY FOUND. GIVE CHECK MARKS FOR THE I-1)-CUBE 0489
   15 IF(JC(I,3).GT.0) GO TO 14                                          0490
      JC(I,3)=1                                                          0491
   14 IF(JC(J,3).GT.0) GO TO 11                                          0492
      JC(J,3)=1                                                          0493
      GO TO 11                                                           0494
   19 CONTINUE                                                           0495
   17 IR=IR+1                                                            0496
      IC(IR,1)=JC(I,1)                                                   0497
      IC(IR,2)=L1                                                        0498
      IF((JC(I,3).EQ.2).AND.(JC(J,3).EQ.2)) GO TO 21                     0499
      IC(IR,3)=0                                                         0500
      IF(JC(I,3).GT.0) GO TO 20                                          0501
      JC(I,3)=1                                                          0502
   20 IF(JC(J,3).GT.0) GO TO 11                                          0503
      JC(J,3)=1                                                          0504
      GO TO 11                                                           0505
   21 JC(IR,3)=2                                                         0506
      GO TO 11                                                           0507
   10 CONTINUE                                                           0508
C........TO FILL MV(I,1).                                                0509
      N=M                                                                0510
      DO 22 I=1,JR                                                       0511
      IF(JC(I,3).GT.0) GO TO 22                                          0512
      N=N+1                                                              0513
      MV(N,1)=JC(I,1)                                                    0514
      MV(N,2)=JC(I,2)                                                    0515
   22 CONTINUE                                                           0516
      RETURN                                                             0517
   25 IF(JC(I,3).EQ.2) GO TO 26                                          0518
      N=M+1                                                              0519
      MV(N,1)=JC(I,1)                                                    0520
      MV(N,2)=JC(I,2)                                                    0521
   26 RETURN                                                             0522
      END                                                                0523
C........GIVEN, MW(0,1,2); FIND MV  ALL P1'S .                           0524
      SUBROUTINE MVFMW(NN,N2N,NN2,NV,MW,MK,M2,JC,IC,MV,N)                0525
*     INTEGER*2 MW(NN),MK(NN),M2(NV),JC(N2N,3),IC(N2N,3),MV(NN2,2)       0526
      CALL GEN1C(NN,N2N,NN2,NV,MW,MK,M2,IC,IR,MV,N)                      0527
   10 IF(IR.EQ.0) GO TO 30                                               0528
      M=N                                                                0529
      CALL ICFJC(NN,N2N,NN2,NV,M2,IC,IR,JC,JR,M,MV,N)                    0530
                                                                         0531
```

```
      IF(JR.EQ.0) GO TO 30                                              0532
      M=N                                                               0533
      CALL ICFJC(NN,N2N,NN2,NV,M2,JC,JR,IC,IR,M,MV,N)                   0534
      GO TO 20                                                          0535
   30 RETURN                                                            0536
      END                                                               0537
C.....SUBROUTINE TO MODIFY MT USING MK AFTER COVERING.                  0538
      SUBROUTINE MODMT(NN,NF,K,MT,MK)                                   0539
*     INTEGER*2 MT(NN,NF),MK(NN)                                        0540
      DO 10 I=1,NN                                                      0541
      IF(MK(I).EQ.0) GO TO 10                                           0542
      MT(I,K)=1                                                         0543
   10 CONTINUE                                                          0544
      RETURN                                                            0545
      END                                                               0546
*     SUBROUTINE SLCT(NN,NN2,NV,N,MV,MVK,MW,MK,M2,MCVR,LNPI,K1,K2,K3,K4,0547
*    1K5,K6,K7,K8,MST)                                                  0548
*     INTEGER*2 MCVR(NN,NN2),MV(NN2,2),MVK(NN2),MW(NN),MK(NN),MST(48,10)0549
      DO 10 I=1,NN                                                      0550
   10 MK(I)=0                                                           0551
      CALL CNTNX(NN,MW,NX)                                              0552
      IF(NX.EQ.0) GO TO 19                                              0553
      CALL CVRFL(NN,NN2,M2,NV,N,MV,MCVR)                                0554
      DO 11 J=1,N                                                       0555
   11 MVK(J)=0                                                          0556
      CALL EPIDT(NN,N,MW,MK,MCVR,MVK)                                   0557
      CALL CNTIX(NN,MW,MK,IXL)                                          0558
      IF(IXL.EQ.0) GO TO 16                                             0559
   13 CALL COLDM(NN,N,MW,MK,MCVR,MVK)                                   0560
      CALL ROWDM(NN,N,MW,MK,MCVR,MVK)                                   0561
      CALL EPIDT(NN,N,MW,MK,MCVR,MVK)                                   0562
      IX=IXL                                                            0563
   14 CALL CNTIX(NN,MW,MK,IXL)                                          0564
      IF(IXL.EQ.IX) GO TO 15                                            0565
      IF(IXL.EQ.0) GO TO 16                                             0566
      GO TO 13                                                          0567
   15 WRITE(6,102)                                                      0568
  102 FORMAT(/,10X,'CYCLIC CASE. APPROXIMATE MINIMUM SOLUTION IS OBTAINE0569
     1D.',/)                                                            0570
      CALL CYCLC(NN,N,MK,MCVR,MVK)                                      0571
      GO TO 14                                                          0572
   16 DO 17 J=1,N                                                       0573
      IF(MVK(J).NE.1) GO TO 17                                          0574
*     LNPI=LNPI+1                                                       0575
      WRITE(6,101)MV(J,1),MV(J,2)                                       0576
*     IF(LNPI.GT.48) GO TO 17                                           0577
*     MST(LNPI,1)=MV(J,1)                                               0578
*     MST(LNPI,2)=MV(J,2)                                               0579
*     MST(LNPI,3)=K1                                                    0580
*     MST(LNPI,4)=K2                                                    0581
*     MST(LNPI,5)=K3                                                    0582
*     MST(LNPI,6)=K4                                                    0583
*     MST(LNPI,7)=K5                                                    0584
*     MST(LNPI,8)=K6                                                    0585
*     MST(LNPI,9)=K7                                                    0586
*     MST(LNPI,10)=K8                                                   0587
   17 CONTINUE                                                          0588
  101 FORMAT(20X,'(',I5,1X,',',I6,')')                                  0589
      RETURN                                                            0590
```

```
   19 WRITE(6,106)                                                      0591
  106 FORMAT(/,10X,'*** NO EXCLUSIVE MINTERMS TO BE COVERED AT THIS STEP0592
     1***',/)                                                           0593
      RETURN                                                            0594
C                                                                       0595
      END                                                               0596
      SUBROUTINE COLDM(NN,N,MW,MK,MCVR,MVK)                             0597
*     INTEGER*2 MCVR(NN,N),MVK(N),MW(NN),MK(NN)                         0598
      I=1                                                               0599
   13 IF((MW(I).EQ.1).AND.(MK(I).EQ.0)) GO TO 16                        0600
   10 I=I+1                                                             0601
      IF(I.EQ.NN) GO TO 25                                              0602
      GO TO 13                                                          0603
   16 L=I+1                                                             0604
   14 IF((MW(L).EQ.1).AND.(MK(I).EQ.0)) GO TO 17                        0605
   15 IF(L.EQ.NN) GO TO 10                                              0606
      L=L+1                                                             0607
      GO TO 14                                                          0608
   17 NI=0                                                              0609
      NL=0                                                              0610
      DO 11 J=1,N                                                       0611
      IF(MVK(J).GT.0) GO TO 11                                          0612
      IF(MCVR(I,J)-MCVR(L,J)) 18,11,19                                  0613
   18 NL=NL+1                                                           0614
      GO TO 11                                                          0615
   19 NI=NI+1                                                           0616
   11 CONTINUE                                                          0617
      IF((NI.EQ.0).AND.(NL.EQ.0)) GO TO 20                              0618
      IF((NI.GT.0).AND.(NL.GT.0)) GO TO 15                              0619
      IF(NL.EQ.0) GO TO 20                                              0620
      MW(L)=2                                                           0621
      GO TO 15                                                          0622
   20 MW(I)=2                                                           0623
      GO TO 10                                                          0624
   25 RETURN                                                            0625
      END                                                               0626
      SUBROUTINE ROWDM(NN,N,MW,MK,MCVR,MVK)                             0627
*     INTEGER*2 MCVR(NN,N),MVK(N),MW(NN),MK(NN)                         0628
      J=1                                                               0629
   17 IF(MVK(J).EQ.0) GO TO 16                                          0630
   10 J=J+1                                                             0631
      IF(J.EQ.N) GO TO 25                                               0632
      GO TO 17                                                          0633
   16 L=J+1                                                             0634
   18 IF(MVK(L).EQ.0) GO TO 19                                          0635
   15 IF(L.EQ.N) GO TO 10                                               0636
      L=L+1                                                             0637
      GO TO 18                                                          0638
   19 NJ=0                                                              0639
      NL=0                                                              0640
      DO 11 I=1,NN                                                      0641
      IF((MW(I).EQ.1).AND.(MK(I).EQ.0)) GO TO 20                        0642
      GO TO 11                                                          0643
   20 IF(MCVR(I,J)-MCVR(I,L))21,11,22                                   0644
   21 NL=NL+1                                                           0645
      GO TO 11                                                          0646
   22 NJ=NJ+1                                                           0647
   11 CONTINUE                                                          0648
      IF((NL.EQ.0).AND.(NJ.EQ.0)) GO TO 23                              0649
```

```
      IF((NL.GT.0).AND.(NJ.GT.0)) GO TO 15                              0650
      IF(NJ.EQ.0) GO TO 23                                              0651
      MVK(L)=2                                                          0652
      GO TO 25                                                          0653
   23 MVK(J)=2                                                          0654
      GO TO 10                                                          0655
   25 RETURN                                                            0656
      END                                                               0657
      SUBROUTINE EPIDT(NN,N,MW,MK,MCVR,MVK)                             0658
*     INTEGER*2 MCVR(NN,N),MVK(N),MW(NN),MK(NN)                         0659
      I=1                                                               0660
   10 IF((MW(I).EQ.1).AND.(MK(I).EQ.0)) GO TO 12                        0661
   11 IF(I.EQ.NN) GO TO 21                                              0662
      I=I+1                                                             0663
      GO TO 10                                                          0664
   12 M=0                                                               0665
      J=1                                                               0666
   13 IF(MVK(J).GT.0) GO TO 14                                          0667
      IF(MCVR(I,J).EQ.1) GO TO 15                                       0668
   14 IF(J.EQ.N) GO TO 16                                               0669
      J=J+1                                                             0670
      GO TO 13                                                          0671
   15 M=M+1                                                             0672
      L=J                                                               0673
      GO TO 14                                                          0674
   16 IF(M.EQ.0) GO TO 17                                               0675
      IF(M.EQ.1) GO TO 18                                               0676
      GO TO 11                                                          0677
   17 MW(I)=2                                                           0678
      GO TO 11                                                          0679
   18 MVK(L)=1                                                          0680
      DO 20 II=1,NN                                                     0681
      IF(MCVR(II,L).EQ.1) GO TO 19                                      0682
      GO TO 20                                                          0683
   19 MK(II)=1                                                          0684
   20 CONTINUE                                                          0685
      GO TO 11                                                          0686
   21 RETURN                                                            0687
      END                                                               0688
      SUBROUTINE CNTIX(NN,MW,MK,IXL)                                    0689
*     INTEGER*2 MW(NN),MK(NN)                                           0690
      IXL=0                                                             0691
      DO 11 I=1,NN                                                      0692
      IF((MW(I).EQ.1).AND.(MK(I).EQ.0)) GO TO 10                        0693
      GO TO 11                                                          0694
   10 IXL=IXL+1                                                         0695
   11 CONTINUE                                                          0696
      RETURN                                                            0697
      END                                                               0698
      SUBROUTINE CYCLC(NN,N,MK,MCVR,MVK)                                0699
*     INTEGER*2 MCVR(NN,N),MVK(N),MK(NN)                                0700
      J=N                                                               0701
   10 IF(MVK(J).EQ.0) GO TO 12                                          0702
      J=J-1                                                             0703
      IF(J.EQ.0) GO TO 18                                               0704
      GO TO 10                                                          0705
   12 DO 17 I=1,NN                                                      0706
      IF(MCVR(I,J).EQ.1) GO TO 16                                       0707
      GO TO 17                                                          0708
```

```
      16 MK(I)=1                                                        0709
      17 CONTINUE                                                       0710
      18 RETURN                                                         0711
         END                                                            0712
         SUBROUTINE ARANG(NN,NF,MT)                                     0713
*        INTEGER*2 MT(NN,NF)                                            0714
         DO 19 J=1,NF                                                   0715
         DO 19 I=1,NN                                                   0716
         IF(MT(I,J).NE.1) GO TO 19                                      0717
         MT(I,J)=0                                                      0718
      19 CONTINUE                                                       0719
         RETURN                                                         0720
         END                                                            0721
C                                                                       0722
         SUBROUTINE MTFIL(NN,NF,MT,MW,L)                                0723
*        INTEGER*2 MT(NN,NF),MW(NN)                                     0724
         DO 9 K=1,NF                                                    0725
         DO 9 I=1,NN                                                    0726
       9 MT(I,K)=0                                                      0727
         DO 10 K=1,NF                                                   0728
         WRITE(7,2)                                                     0729
       2 FORMAT(/,5X,'ENTER; FUNCTION NO., NO.OF MINTERMS, NO.OF DCNT-CARES0730
        1')                                                             0731
         READ(5,*)K1,K2,K3                                              0732
         WRITE(6,100)K1,K2,K3                                           0733
         K4=K2+1                                                        0734
         WRITE(7,3)                                                     0735
       3 FORMAT(/,5X,'ENTER; MINTERMS, -1')                             0736
         READ(5,*)(MW(I),I=1,K4)                                        0737
         WRITE(6,101)(MW(I),I=1,K2)                                     0738
         IF(MW(K4).GT.0) GO TO 13                                       0739
         L=1                                                            0740
         K5=K2-K3                                                       0741
         DO 11 I=1,K5                                                   0742
         J=MW(I)+1                                                      0743
      11 MT(J,K)=2                                                      0744
         IF(K3.EQ.0) GO TO 10                                           0745
         DO 12 I=1,K3                                                   0746
         J=MW(I+K5)+1                                                   0747
      12 MT(J,K)=1                                                      0748
      10 CONTINUE                                                       0749
     100 FORMAT(10X,'***FUNCTION-',I2,/,10X,'NO. OF ALL MINTERMS=',I5,  0750
        1/,10X,'NO. OF DONT-CARES=',I5,/)                               0751
     101 FORMAT(/,37X,'MINTERMS:',//,30(17X,10I5,/))                    0752
         RETURN                                                         0753
      13 WRITE(6,102)K                                                  0754
     102 FORMAT (10X,5('* '),'ERROR IN DATA OF FUNCTION-',I2,5(' *'),/) 0755
         WRITE(6,103)(MW(I),I=1,K4)                                     0756
     103 FORMAT(30(17X,10I5,/))                                         0757
         L=0                                                            0758
         RETURN                                                         0759
         END                                                            0760
C.........SUBROUTINE TO FIND NO. OF YET UNCOVERED MINTERMS=NG           0761
         SUBROUTINE FNDNG(NN,NF,MT,NG)                                  0762
*        INTEGER*2 MT(NN,NF)                                            0763
         NG=0                                                           0764
         DO 20 I=1,NN                                                   0765
         DO 16 K=1,NF                                                   0766
         IF(MT(I,K).EQ.2) GO TO 15                                      0767
```

```
   10 CONTINUE                                                          0768
      GO TO 20                                                          0769
   15 NG=NG+1                                                           0770
   20 CONTINUE                                                          0771
      RETURN                                                            0772
      END                                                               0773
C.....SUBROUTINE TO FIND MW(C,1,2) FOR PI DETERMINATION & LEVERING      0774
C.....                                                                  0775
C.....                                                                  0776
C.....N=1 FOR SINGLE FUNCTION, 2 FOR TWO FUNCTIONS, ... 7=FOR          0777
C.....PRODUCT OF 1-FUNCTIONS.                                           0778
      SUBROUTINE MWPMT(NN,NF,NV,K,MT,MW,N,L)                            0779
*     INTEGER*2 MT(NN,NF),MW(NN)                                        0780
      DO 10 I=1,NN                                                      0781
      M=N                                                               0782
      IF(MT(I,K).LE.L) GO TO 13                                         0783
      DO 11 J=1,NF                                                      0784
      IF(J.EQ.K) GO TO 11                                               0785
      IF(MT(I,J).LE.L) GO TO 11                                         0786
      M=M+1                                                             0787
   11 CONTINUE                                                          0788
      IF(M.GT.N) GO TO 12                                               0789
      MW(I)=1                                                           0790
      GO TO 10                                                          0791
   12 MW(I)=2                                                           0792
      GO TO 10                                                          0793
   13 MW(I)=0                                                           0794
   10 CONTINUE                                                          0795
      RETURN                                                            0796
      END                                                               0797
C .....MW2MT FOR 2-FUNCTIONS                                            0798
      SUBROUTINE MW2MT(NN,NF,NV,K1,K2,MT,MW,N,L)                        0799
*     INTEGER*2 MT(NN,NF),MW(NN)                                        0800
      DO 10 I=1,NN                                                      0801
      M=N                                                               0802
      IF(MT(I,K1).LE.L) GO TO 13                                        0803
      IF(MT(I,K2).LE.L) GO TO 13                                        0804
      DO 11 J=1,NF                                                      0805
      IF(J.EQ.K1) GO TO 11                                              0806
      IF(J.EQ.K2) GO TO 11                                              0807
      IF(MT(I,J).LE.L) GO TO 11                                         0808
      M=M+1                                                             0809
   11 CONTINUE                                                          0810
      IF(M.GT.N) GO TO 12                                               0811
      MW(I)=1                                                           0812
      GO TO 10                                                          0813
   12 MW(I)=2                                                           0814
      GO TO 10                                                          0815
   13 MW(I)=0                                                           0816
   10 CONTINUE                                                          0817
      RETURN                                                            0818
      END                                                               0819
C .....MW3MT FOR 3-FUNCTIONS                                            0820
      SUBROUTINE MW3MT(NN,NF,NV,K1,K2,K3,MT,MW,N,L)                     0821
*     INTEGER*2 MT(NN,NF),MW(NN)                                        0822
      DO 10 I=1,NN                                                      0823
      M=N                                                               0824
      IF(MT(I,K1).LE.L) GO TO 13                                        0825
      IF(MT(I,K2).LE.L) GO TO 13                                        0826
```

```
      IF(MT(I,K3).LE.L) GO TO 13                                        0827
      DO 11 J=1,NF                                                      0828
      IF(J.EQ.K1) GO TO 11                                              0829
      IF(J.EQ.K2) GO TO 11                                              0830
      IF(J.EQ.K3) GO TO 11                                              0831
      IF(MT(I,J).LE.L) GO TO 11                                         0832
      M=M+1                                                             0833
   11 CONTINUE                                                          0834
      IF(M.GT.N) GO TO 12                                               0835
      MW(I)=1                                                           0836
      GO TO 10                                                          0837
   12 MW(I)=2                                                           0838
      GO TO 10                                                          0839
   13 MW(I)=0                                                           0840
   10 CONTINUE                                                          0841
      RETURN                                                            0842
      END                                                               0843
C .......MW4MT FOR 4-FUNCTIONS                                          0844
      SUBROUTINE MW4MT(NN,NF,NV,K1,K2,K3,K4,MT,MW,N,L)                  0845
*     INTEGER*2 MT(NN,NF),MW(NN)                                        0846
      DO 10 I=1,NN                                                      0847
      M=N                                                               0848
      IF(MT(I,K1).LE.L) GO TO 13                                        0849
      IF(MT(I,K2).LE.L) GO TO 13                                        0850
      IF(MT(I,K3).LE.L) GO TO 13                                        0851
      IF(MT(I,K4).LE.L) GO TO 13                                        0852
      DO 11 J=1,NF                                                      0853
      IF(J.EQ.K1) GO TO 11                                              0854
      IF(J.EQ.K2) GO TO 11                                              0855
      IF(J.EQ.K3) GO TO 11                                              0856
      IF(J.EQ.K4) GO TO 11                                              0857
      IF(MT(I,J).LE.L) GO TO 11                                         0858
      M=M+1                                                             0859
   11 CONTINUE                                                          0860
      IF(M.GT.N) GO TO 12                                               0861
      MW(I)=1                                                           0862
      GO TO 10                                                          0863
   12 MW(I)=2                                                           0864
      GO TO 10                                                          0865
   13 MW(I)=0                                                           0866
   10 CONTINUE                                                          0867
      RETURN                                                            0868
      END                                                               0869
C .......MW5MT FOR 5-FUNCTIONS                                          0870
      SUBROUTINE MW5MT(NN,NF,NV,K1,K2,K3,K4,K5,MT,MW,N,L)               0871
*     INTEGER*2 MT(NN,NF),MW(NN)                                        0872
      DO 10 I=1,NN                                                      0873
      M=N                                                               0874
      IF(MT(I,K1).LE.L) GO TO 13                                        0875
      IF(MT(I,K2).LE.L) GO TO 13                                        0876
      IF(MT(I,K3).LE.L) GO TO 13                                        0877
      IF(MT(I,K4).LE.L) GO TO 13                                        0878
      IF(MT(I,K5).LE.L) GO TO 13                                        0879
      DO 11 J=1,NF                                                      0880
      IF(J.EQ.K1) GO TO 11                                              0881
      IF(J.EQ.K2) GO TO 11                                              0882
      IF(J.EQ.K3) GO TO 11                                              0883
      IF(J.EQ.K4) GO TO 11                                              0884
      IF(J.EQ.K5) GO TO 11                                              0885
```

```
      IF(MT(I,J).LE.L) GO TO 11                                          0886
      M=M+1                                                              0887
   11 CONTINUE                                                           0888
      IF(M.GT.N) GO TO 12                                                0889
      MW(I)=1                                                            0890
      GO TO 10                                                           0891
   12 MW(I)=2                                                            0892
      GO TO 10                                                           0893
   13 MW(I)=0                                                            0894
   10 CONTINUE                                                           0895
      RETURN                                                             0896
      END                                                                0897
C.....MW6MT FOR 6-FUNCTIONS                                              0898
      SUBROUTINE MW6MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,MT,MW,N,L)             0899
*     INTEGER*2 MT(NN,NF),MW(NN)                                         0900
      DO 10 I=1,NN                                                       0901
      M=N                                                                0902
      IF(MT(I,K1).LE.L) GO TO 13                                         0903
      IF(MT(I,K2).LE.L) GO TO 13                                         0904
      IF(MT(I,K3).LE.L) GO TO 13                                         0905
      IF(MT(I,K4).LE.L) GO TO 13                                         0906
      IF(MT(I,K5).LE.L) GO TO 13                                         0907
      IF(MT(I,K6).LE.L) GO TO 13                                         0908
      DO 11 J=1,NF                                                       0909
      IF(J.EQ.K1) GO TO 11                                               0910
      IF(J.EQ.K2) GO TO 11                                               0911
      IF(J.EQ.K3) GO TO 11                                               0912
      IF(J.EQ.K4) GO TO 11                                               0913
      IF(J.EQ.K5) GO TO 11                                               0914
      IF(J.EQ.K6) GO TO 11                                               0915
      IF(MT(I,J).LE.L) GO TO 11                                          0916
      M=M+1                                                              0917
   11 CONTINUE                                                           0918
      IF(M.GT.N) GO TO 12                                                0919
      MW(I)=1                                                            0920
      GO TO 10                                                           0921
   12 MW(I)=2                                                            0922
      GO TO 10                                                           0923
   13 MW(I)=0                                                            0924
   10 CONTINUE                                                           0925
      RETURN                                                             0926
      END                                                                0927
C.....MW7MT FOR 7-FUNCTIONS                                              0928
      SUBROUTINE MW7MT(NN,NF,NV,K1,K2,K3,K4,K5,K6,K7,MT,MW,N,L)          0929
*     INTEGER*2 MT(NN,NF),MW(NN)                                         0930
      DO 10 I=1,NN                                                       0931
      M=N                                                                0932
      IF(MT(I,K1).LE.L) GO TO 13                                         0933
      IF(MT(I,K2).LE.L) GO TO 13                                         0934
      IF(MT(I,K3).LE.L) GO TO 13                                         0935
      IF(MT(I,K4).LE.L) GO TO 13                                         0936
      IF(MT(I,K5).LE.L) GO TO 13                                         0937
      IF(MT(I,K6).LE.L) GO TO 13                                         0938
      IF(MT(I,K7).LE.L) GO TO 13                                         0939
      DO 11 J=1,NF                                                       0940
      IF(J.EQ.K1) GO TO 11                                               0941
      IF(J.EQ.K2) GO TO 11                                               0942
      IF(J.EQ.K3) GO TO 11                                               0943
      IF(J.EQ.K4) GO TO 11                                               0944
```

```
      IF(J.EQ.K5) GO TO 11                                              0945
      IF(J.EQ.K6) GO TO 11                                              0946
      IF(J.EQ.K7) GO TO 11                                              0947
      IF(MT(I,J).LE.L) GO TO 11                                         0948
      M=M+1                                                             0949
   11 CONTINUE                                                          0950
      IF(M.GT.N) GO TO 12                                               0951
      MW(I)=1                                                           0952
      GO TO 10                                                          0953
   12 MW(I)=2                                                           0954
      GO TO 10                                                          0955
   13 MW(I)=0                                                           0956
   10 CONTINUE                                                          0957
      RETURN                                                            0958
      END                                                               0959
C .....MWBMT FOR 8-FUNCTIONS                                            0960
      SUBROUTINE MWBMT(NN,NF,NV,K1,K2,K3,K4,K5,K6,K7,KE,MT,MW,N,L)      0961
*     INTEGER*2 MT(NN,NF),MW(NN)                                        0962
      DO 10 I=1,NN                                                      0963
      M=N                                                               0964
      IF(MT(I,K1).LE.L) GO TO 13                                        0965
      IF(MT(I,K2).LE.L) GO TO 13                                        0966
      IF(MT(I,K3).LE.L) GO TO 13                                        0967
      IF(MT(I,K4).LE.L) GO TO 13                                        0968
      IF(MT(I,K5).LE.L) GO TO 13                                        0969
      IF(MT(I,K6).LE.L) GO TO 13                                        0970
      IF(MT(I,K7).LE.L) GO TO 13                                        0971
      IF(MT(I,KE).LE.L) GO TO 13                                        0972
      DO 11 J=1,NF                                                      0973
      IF(J.EQ.K1) GO TO 11                                              0974
      IF(J.EQ.K2) GO TO 11                                              0975
      IF(J.EQ.K3) GO TO 11                                              0976
      IF(J.EQ.K4) GO TO 11                                              0977
      IF(J.EQ.K5) GO TO 11                                              0978
      IF(J.EQ.K6) GO TO 11                                              0979
      IF(J.EQ.K7) GO TO 11                                              0980
      IF(J.EQ.K8) GO TO 11                                              0981
      IF(MT(I,J).LE.L) GO TO 11                                         0982
      M=M+1                                                             0983
   11 CONTINUE                                                          0984
      IF(M.GT.N) GO TO 12                                               0985
      MW(I)=1                                                           0986
      GO TO 10                                                          0987
   12 MW(I)=2                                                           0988
      GO TO 10                                                          0989
   13 MW(I)=0                                                           0990
   10 CONTINUE                                                          0991
      RETURN                                                            0992
      END                                                               0993
      SUBROUTINE CVRFL(NN,NN2,M2,NV,N,MV,MCVR)                          0994
*     INTEGER*2 MV(NN2,2),MCVR(NN,NN2),M2(NV)                           0995
      DO 9 J=1,N                                                        0996
      DO 9 I=1,NN                                                       0997
    9 MCVR(I,J)=0                                                       0998
      DO 10 J=1,N                                                       0999
      II=MV(J,1)+1                                                      1000
      MCVR(II,J)=1                                                      1001
      IF(MV(J,2).EQ.0) GO TO 10                                         1002
      I=II+MV(J,2)                                                      1003
```

```
      MCVR(I,J)=1                                                       1004
      M=MV(J,2)                                                         1005
      LL=0                                                              1006
      DO 11 K=1,NV                                                      1007
      L=2**(NV-K)                                                       1008
      IF(M-L)11,12,12                                                   1009
   12 LL=LL+1                                                           1010
      M=M-L                                                             1011
      M2(LL)=L                                                          1012
   11 CONTINUE                                                          1013
      DO 20 K=1,LL                                                      1014
   20 MCVR(II+M2(K)),J)=1                                               1015
      IF(LL.LT.2) GO TO 10                                              1016
      I=II+M2(1)+M2(2)                                                  1017
      MCVR(I,J)=1                                                       1018
      IF(LL.LT.3) GO TO 10                                              1019
      N2=LL-1                                                           1020
      DO 13 J1=1,N2                                                     1021
      K2=J1+1                                                           1022
      DO 13 J2=K2,LL                                                    1023
      I=II+M2(J1)+M2(J2)                                                1024
      MCVR(I,J)=1                                                       1025
   13 CONTINUE                                                          1026
      IF(LL.LT.4) GO TO 10                                              1027
      N3=LL-2                                                           1028
      DO 14 J1=1,N3                                                     1029
      K2=J1+1                                                           1030
      DO 14 J2=K2,N2                                                    1031
      K3=J2+1                                                           1032
      DO 14 J3=K3,LL                                                    1033
      I=II+M2(J1)+M2(J2)+M2(J3)                                         1034
   14 MCVR(I,J)=1                                                       1035
      IF(LL.LT.5) GO TO 10                                              1036
      N4=LL-3                                                           1037
      DO 15 J1=1,N4                                                     1038
      K2=J1+1                                                           1039
      DO 15 J2=K2,N3                                                    1040
      K3=J2+1                                                           1041
      DO 15 J3=K3,N2                                                    1042
      K4=J3+1                                                           1043
      DO 15 J4=K4,LL                                                    1044
      I=II+M2(J1)+M2(J2)+M2(J3)+M2(J4)                                  1045
   15 MCVR(I,J)=1                                                       1046
      IF(LL.LT.6) GO TO 10                                              1047
      N5=LL-4                                                           1048
      DO 16 J1=1,N5                                                     1049
      K2=J1+1                                                           1050
      DO 16 J2=K2,N4                                                    1051
      K3=J2+1                                                           1052
      DO 16 J3=K3,N3                                                    1053
      K4=J3+1                                                           1054
      DO 16 J4=K4,N2                                                    1055
      K5=J4+1                                                           1056
      DO 16 J5=K5,LL                                                    1057
      I=II+M2(J1)+M2(J2)+M2(J3)+M2(J4)+M2(J5)                           1058
   16 MCVR(I,J)=1                                                       1059
      IF(LL.LT.7) GO TO 10                                              1060
      N6=LL-5                                                           1061
      DO 17 J1=1,N6                                                     1062
```

```
      K2=J1+1                                                          1063
      DO 17 J2=K2,N5                                                   1064
      K3=J2+1                                                          1065
      DO 17 J3=K3,N4                                                   1066
      K4=J3+1                                                          1067
      DO 17 J4=K4,N3                                                   1068
      K5=J4+1                                                          1069
      DO 17 J5=K5,N2                                                   1070
      K6=J5+1                                                          1071
      DO 17 J6=K6,LL                                                   1072
      I=I1+M2(J1)+M2(J2)+M2(J3)+M2(J4)+M2(J5)+M2(J6)                   1073
   17 MCVR(1,J)=1                                                      1074
      IF(LL.LT.6) GO TO 10                                             1075
      K7=LL-6                                                          1076
      DO 18 J1=1,N7                                                    1077
      K2=J1+1                                                          1078
      DO 18 J2=K2,N6                                                   1079
      K3=J2+1                                                          1080
      DO 18 J3=K3,N5                                                   1081
      K4=J3+1                                                          1082
      DO 18 J4=K4,N4                                                   1083
      K5=J4+1                                                          1084
      DO 18 J5=K5,N3                                                   1085
      K6=J5+1                                                          1086
      DO 18 J6=K6,N2                                                   1087
      K7=J6+1                                                          1088
      DO 18 J7=K7,LL                                                   1089
      I=I1+M2(J1)+M2(J2)+M2(J3)+M2(J4)+M2(J5)+M2(J6)+M2(J7)            1090
   18 MCVR(1,J)=1                                                      1091
   10 CONTINUE                                                         1092
      RETURN                                                           1093
      END                                                              1094
```